Aplicación Práctica de R

Para el análisis de datos en investigación

Servio Interiano



Aplicación Práctica de R

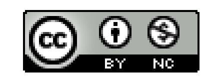
Para el análisis de datos en investigación

Servio Interiano

Coordinador del curso de Estadística
Facultad de Odontología, Universidad de San Carlos de Guatemala

Guatemala 19 de marzo de 2022

Versión electrónica, previo a revisión editorial,



Basado en la plantilla fancybook-template
elaborada con TeXLive 2019
Proyecto los academycos, http://losacademycos.com/



Autopublicado mundialmente
Guatemala, México, Madrid, New York, Buenos Aires.
marzo, 2022.



Dedicatoria: para mis hijas, ustedes me han hecho un padre orgulloso y felíz.

Este libro está dedicado especialmente a Lucía, para quien fue escrito originalmente.

Servio Interiano, marzo 2022. servio.interiano@gmail.com

Índice



1. Introducción

El “R” es un lenguaje de programación para análisis estadístico muy versátil y poderoso y su aplicación se ha popularizado en el ámbito del análisis de datos por lo robusto del lenguaje, lo simple de su manejo y aplicación y su naturaleza multiplataforma, lo que lo hace muy útil, sin importar el sistema operativo que se utilice.
Existen muchos cursos para el aprendizaje del R, tanto en instituciones educativas, como en la web. Sin embargo la mayoría de los cursos profundizan en el aspecto de programación en R y no en su aplicación práctica por parte de un investigador que no desee dedicarse a la programación de código más elaborado.

El presente documento busca dar un enfoque más sencillo y práctico al uso de R para el análisis de datos sin profundizar en los aspectos de programación, para investigaciones pequeñas, estudios de tesis, ejercicios de aprendizaje de bioestadística, resolución de ejercicios para estudiantes que cursan estadística y para el análisis de datos en el ámbito docente.

Primeros pasos

El presente libro busca ser un manual de aplicación práctica de R, y no debería ser considerado un curso introductorio al mismo. Es muy recomendable que se reciba un curso inicial para conocer qué es el R, su historia, su naturaleza, su aplicación como lenguaje de programación y su empleo en el análisis de datos.

Asumo que ya existe un conocimiento básico previo y haré énfasis en explicar algunos pasos prácticos para el uso del R desde la consola de comandos, para que las instrucciones sean aplicables en cualquier sistema operativo o plataforma.

Es recomendable tener otros libros de referencia para profundizar en los comandos, programación, sintaxis, etc. así como blogs y publicaciones de páginas web que puedan orientarnos.

Dos textos indispensables, que se encuentran disponibles en la web son:

- “R para principiantes” de Emmanuel Paradis.
- “Introducción a R Notas sobre R: Un entorno de programación para Análisis de Datos y Gráficos” Versión 1.0.1 (2000-05-16) de R Core Team.

Existen algunas diferencias fundamentales en el manejo de los distintos sistemas operativos que deben tenerse en cuenta, en Windows los directorios se manejan de manera diferente a como se hace en Linux, de tal manera que el directorio raíz de Windows probablemente sea “C:\” mientras que en Linux probablemente iniciemos en “home/usuario/” por lo que el archivo de trabajo va a ser diferente dependiendo del sistema operativo que se utilice.

En lo personal mi sistema operativo es el Linux desde hace muchos años, por lo que voy a presentar el manejo del R desde la consola de Linux.
Sin embargo el manejo del R en sí mismo es igual sin importar el sistema operativo o la plataforma que se esté empleando, por lo que el presente manual puede ser aplicado prácticamente sin modificación, excepto en los aspectos que involucren otros directorios.

El uso de R Studio y la ventana de comandos en Windows se cubre en otros cursos existentes en la web y no voy a hacer referencia a ellos.

Algunos comandos para tener en cuenta:

R	, desde la consola, para iniciar el programa R.
q()	, para terminar la sesión y salir de R .
[Ctrl]+[L]	, para borrar la pantalla de trabajo.
Install.packages()	, para descargar paquetes.
library()	, para cargar librerías.
stem()	, produce un diagrama de tallo y hojas.
hist()	, produce un histograma.
Summary()	, produce un resumen de los datos.
table()	, produce un resumen con categorías y recuentos.
head()	, muestra los primeros 6 registros de un objeto.
tail()	, muestra los últimos 6 registros de un objeto.
name()	, muestra los nombres de las variables en un objeto.
ls("package:datasets") base	, muestra el listado de todos los datasets incluidos en R que pueden usarse para prácticas y ejercicios.

Generalidades de R.

R es un lenguaje de programación orientado a objetos, pero podemos utilizarlo para el análisis de datos sin entrar a la parte de programación y elaboración de programas.

Para el usuario avanzado y que desea desatar toda la capacidad de R, es indispensable la programación. Pero para el estudiante, profesor o investigador ocasional, la parte de elaboración de código y programas, es algo que puede dejarse de lado.

Sin embargo, debemos entender algunos términos técnicos para poder comunicarnos de una manera efectiva.

R se ejecuta desde una ventana de comandos, como lo hacían anteriormente los programas que se ejecutaban en mainframes o DOS. Esto es una ventana donde se ingresan comandos en la línea inferior y que da como respuesta una serie de líneas de texto en las líneas superiores como que estuvieran siendo impresas en la pantalla.

Estas líneas que produce el programa como respuesta a nuestros comandos se denominan “salidas” .

Las “salidas” son los resultados que nos interesan y podemos guardarlas en un archivo de texto “salvarlas” o copiarlas directamente con el mouse, como generalmente seleccionamos un texto, con [CTRL] + [C] . Y luego pegarlas con [CTRL]+[v] en nuestro procesador de texto preferido.

Existen ciertos “frentes gráficos” para utilizar con R , como el R Studio, que son programas que interactúan con el usuario para ayudar sobre todo en las tareas de programación.

Para la ejecución de R para análisis de datos, no son necesarios estos programas y hacen poco por el usuario, así que no profundizaré en ellos.

Para iniciar el programa:

- En Windows, hacemos doble click en el ícono de R, o lo buscamos dentro de nuestro menú de programas.
- En Linux, entramos a la consola de comandos, escribimos “R” (sin las comillas) y le damos [Enter]

Una vez dentro de la ventana de R, vamos a observar un signo de “mayor que” , “>” este es el “prompt” desde donde escribimos los comandos y nos indica que el programa está listo para recibir órdenes.

Podemos utilizar el R como una calculadora científica, exactamente de la misma forma que utilizamos una calculadora convencional.

Escribimos las operaciones y el programa nos da la respuesta.

> 2 + 13 15 >	Escribimos la operación y el programa nos da la respuesta.

Podemos guardar los resultados en un objeto denominado “vector”.

Un vector es una colección ordenada de números y es la estructura más simple de datos que es capaz de manejar R.

Pero también podemos guardar una serie de datos o los resultados de operaciones complejas.

> a= 13 + 2 > > a 15 > b= a + 5 > > b 20	Guardamos el resultado de la suma en “a” Pedimos que nos muestre el contenido de “a” Guardamos el resultado de “a + 5” Pedimos que nos muestre el contenido e “b”

Para crear un vector de datos incluimos los mismos en un único objeto a través del comando c() , que podemos entenderlo como “concatenar”.

> serie.a <- c(1,2,3,4,5,) > >serie.a [1] 1, 2, 3, 4, 5 > serie.b = c(6,7,8,9,10)	Ingresamos los datos a un vector de nombre “serie.a”. El “ <- “ simula una flecha, que indica que los datos serán incluidos en el vector. Puede emplearse también el signo igual “=”.



2. Instalación de R

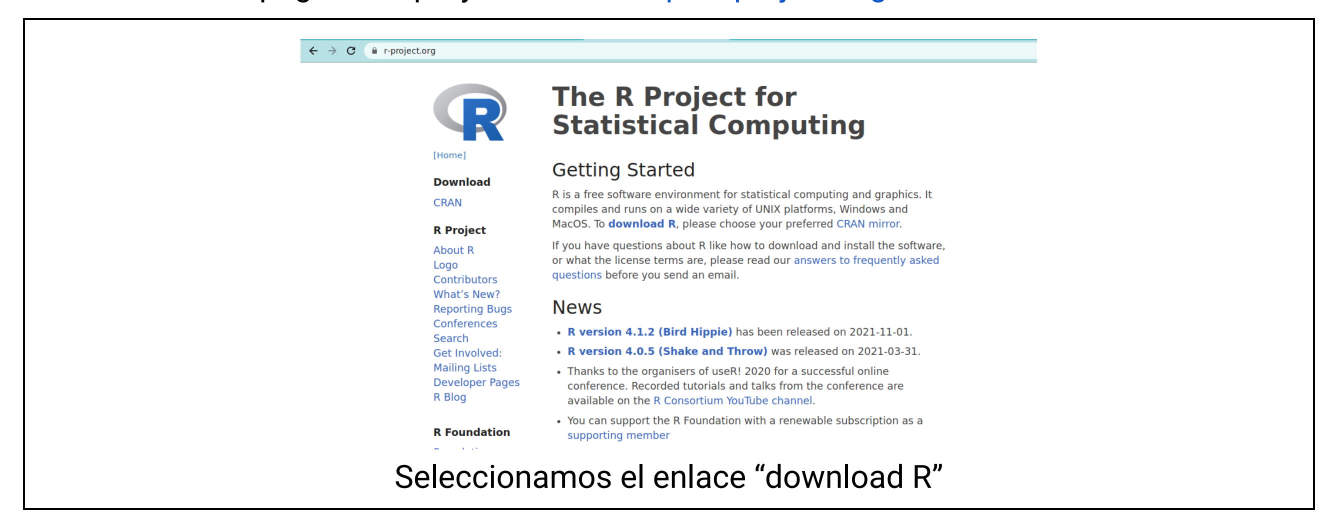
Windows

Para la instalación de R en Windows es recomendable revisar cualquier curso de manejo de R en windows, la mayoría de ellos explica como poder instalar el programa desde el sitio oficial:

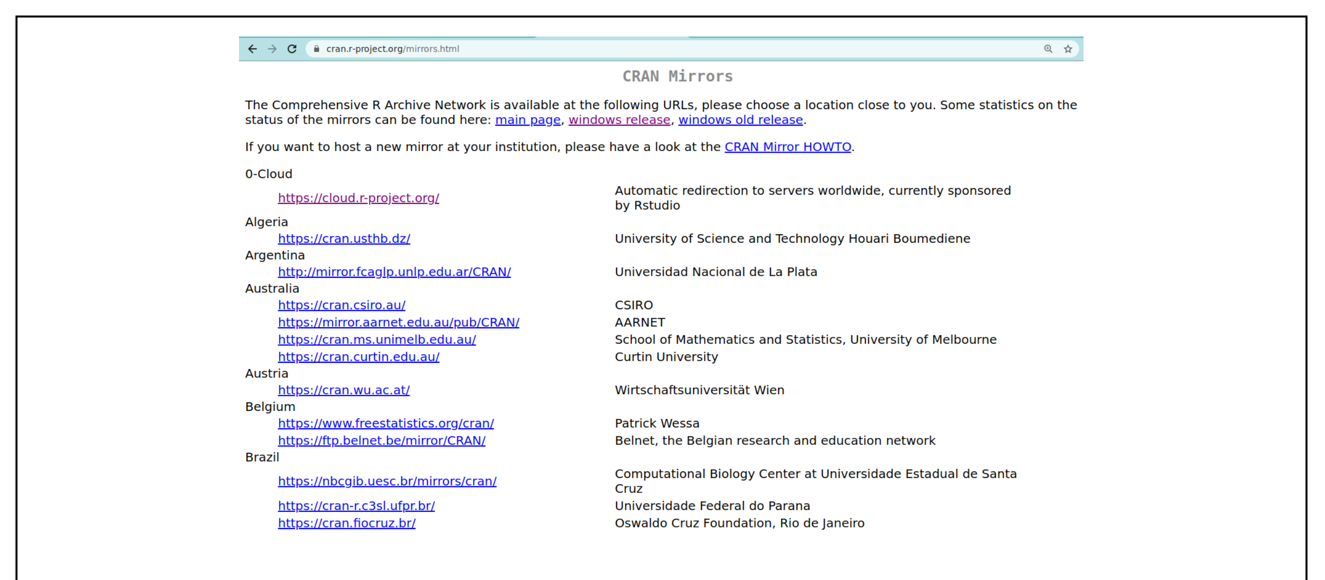
https://www.r-project.org/

Descarga e instalación

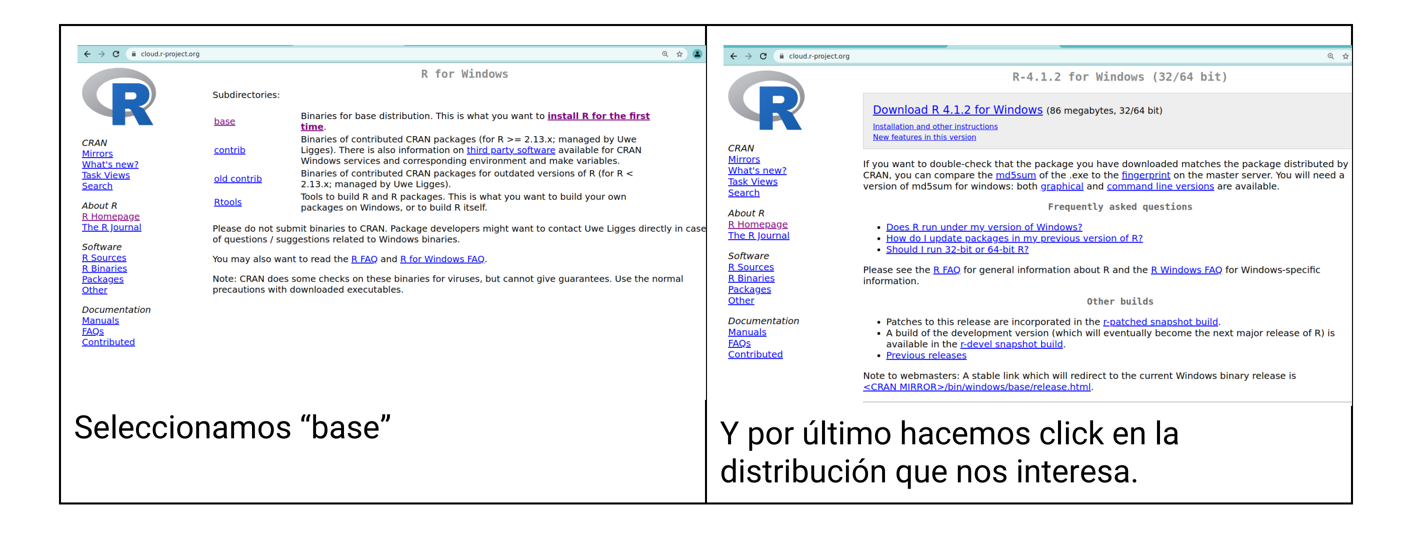
Accedemos a la página del proyecto R en "http://r-project.org"



Seleccionamos el enlace "download R"



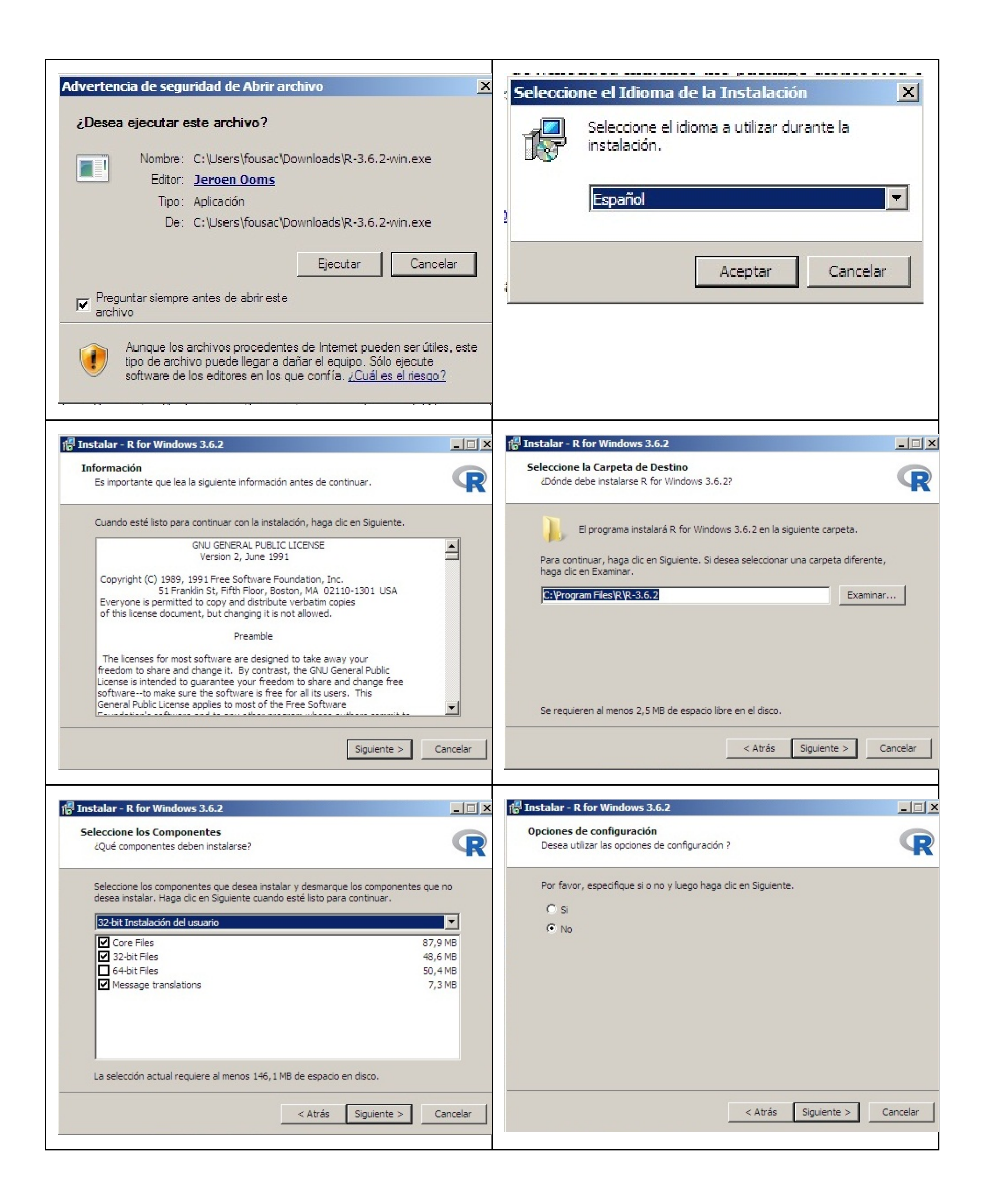
Nos solicita que seleccionemos el servidor que nos convenga. Podemos seleccionar "0-Cloud", o alguno que tengamos en nuestro país de origen.

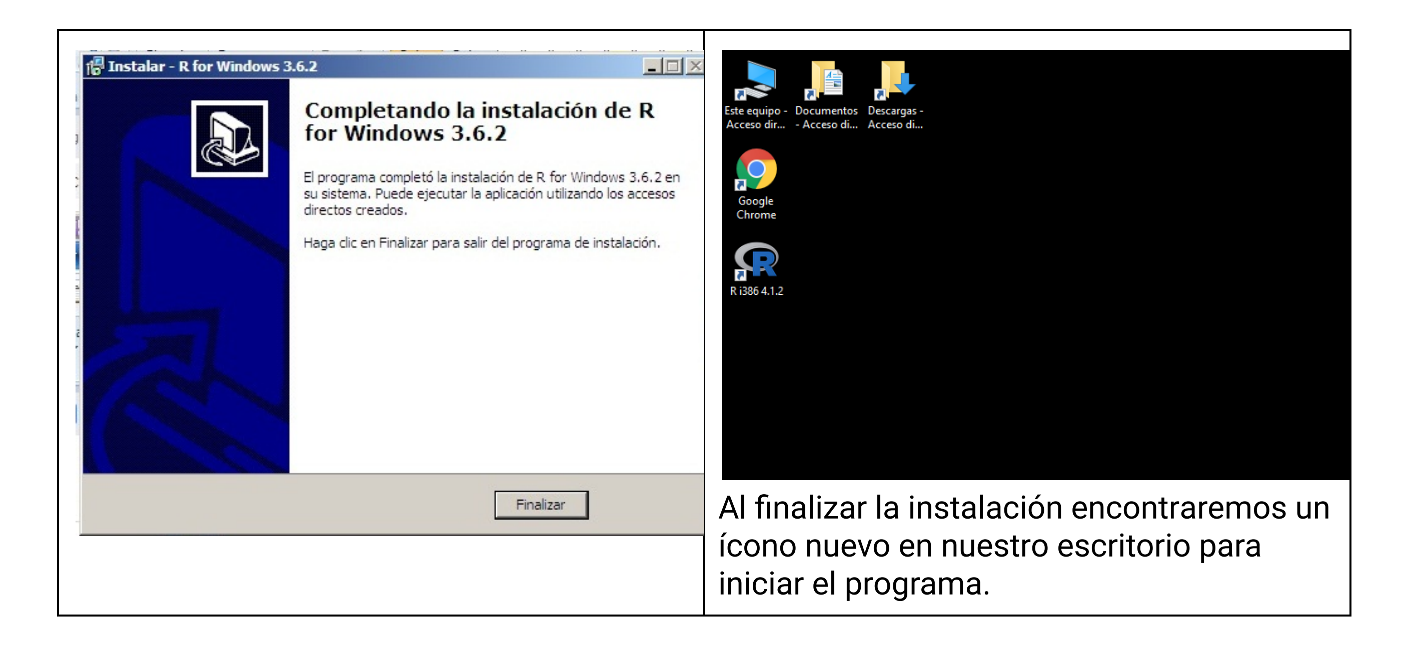


Seleccionamos "base"

Y por último hacemos click en la distribución que nos interesa.

Una vez descargado el paquete, lo abrimos y ejecutamos para instalar en programa.


Advertencia de seguridad de Abrir archivo
¿Desea ejecutar este archivo?
Nombre: C:\Users\fousac\Downloads\R-3.6.2-win.exe
Editor: Jeroen Ooms
Tipo: Aplicación
De: C:\Users\fousac\Downloads\R-3.6.2-win.exe
Ejecutar
Cancelar
Preguntar siempre antes de abrir este archivo
Aunque los archivos procedentes de Internet pueden ser útiles, este tipo de archivo puede llegar a dañar el equipo. Sólo ejecute software de los editores en los que confía. ¿Cuál es el riesgo?
Seleccione el Idioma de la Instalación
Seleccione el idioma a utilizar durante la instalación.
Español
Aceptar
Cancelar
Instalar - R for Windows 3.6.2
Información
Es importante que lea la siguiente información antes de continuar.
Cuando esté listo para continuar con la instalación, haga clic en Siguiente.
GNU GENERAL PUBLIC LICENSE
Version 2, June 1991
Copyright (C) 1989, 1991 Free Software Foundation, Inc.
51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA
Everyone is permitted to copy and distribute verbatim copies
of this license document, but changing it is not allowed.
Preamble
The licenses for most software are designed to take away your
freedom to share and change it. By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users. This
General Public License applies to most of the Free Software
Siguiente >
Cancelar
Instalar - R for Windows 3.6.2
Seleccione la Carpeta de Destino
¿Dónde debe instalarse R for Windows 3.6.2?
El programa instalará R for Windows 3.6.2 en la siguiente carpeta.
Para continuar, haga clic en Siguiente. Si desea seleccionar una carpeta diferente, haga clic en Examinar.
C:\Program Files\R\R-3.6.2
Examinar...
Se requieren al menos 2,5 MB de espacio libre en el disco.
< Atrás
Siguiente >
Cancelar
Instalar - R for Windows 3.6.2
Seleccione los Componentes
¿Qué componentes deben instalarse?
Seleccione los componentes que desea instalar y desmarque los componentes que no desea instalar. Haga clic en Siguiente cuando esté listo para continuar.
32-bit Instalación del usuario
Core Files 87,9 MB
32-bit Files 48,6 MB
64-bit Files 50,4 MB
Message translations 7,3 MB
La selección actual requiere al menos 146,1 MB de espacio en disco.
< Atrás
Siguiente >
Cancelar
Instalar - R for Windows 3.6.2
Opciones de configuración
Desea utilizar las opciones de configuración ?
Por favor, especifique si o no y luego haga clic en Siguiente.
Si
No
< Atrás
Siguiente >
Cancelar



Al finalizar la instalación encontraremos un ícono nuevo en nuestro escritorio para iniciar el programa.

Al ejecutar el programa se nos abrirá una ventana de comandos donde podemos colocar las órdenes y recibir las respuestas.

RGui (32-bit)

Archivo Editar Visualizar Misc Paquetes Ventanas Ayuda

R Console

```
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: i386-w64-mingw32/i386 (32-bit)

R es un software libre y viene sin GARANTIA ALGUNA.
Usted puede redistribuirlo bajo ciertas circunstancias.
Escriba 'license()' o 'licence()' para detalles de distribucion.

R es un proyecto colaborativo con muchos contribuyentes.
Escriba 'contributors()' para obtener más información y
'citation()' para saber cómo citar R o paquetes de R en publicaciones.

Escriba 'demo()' para demostraciones, 'help()' para el sistema on-line de ayuda,
o 'help.start()' para abrir el sistema de ayuda HTML con su navegador.
Escriba 'q()' para salir de R.

>
```

Linux

En la mayoría de distribuciones de Linux se encuentra disponible el paquete desde el repositorio respectivo.
También pueden encontrarse algunas instrucciones desde el sitio oficial (https://www.r-project.org/) .

Dos ejemplos:

- En el caso de que estemos usando Raspbian en una Raspberry Pi, buscamos en el repositorio de la distribución el paquete "r-base" . Esto instala R versión 3.3.3 ARM-Linux de 32 bit. La gran mayoría de distribuciones de Linux funcionan de esta manera.

- En el caso de que estemos usando un Puppy Linux Xenial en cualquier Pc Intel, buscamos en el repositorio propio de la distribución "r-base_3.2.3-4" . Esto instala R versión 3.2.3 i686-pc-linux-gnu (32-bit).

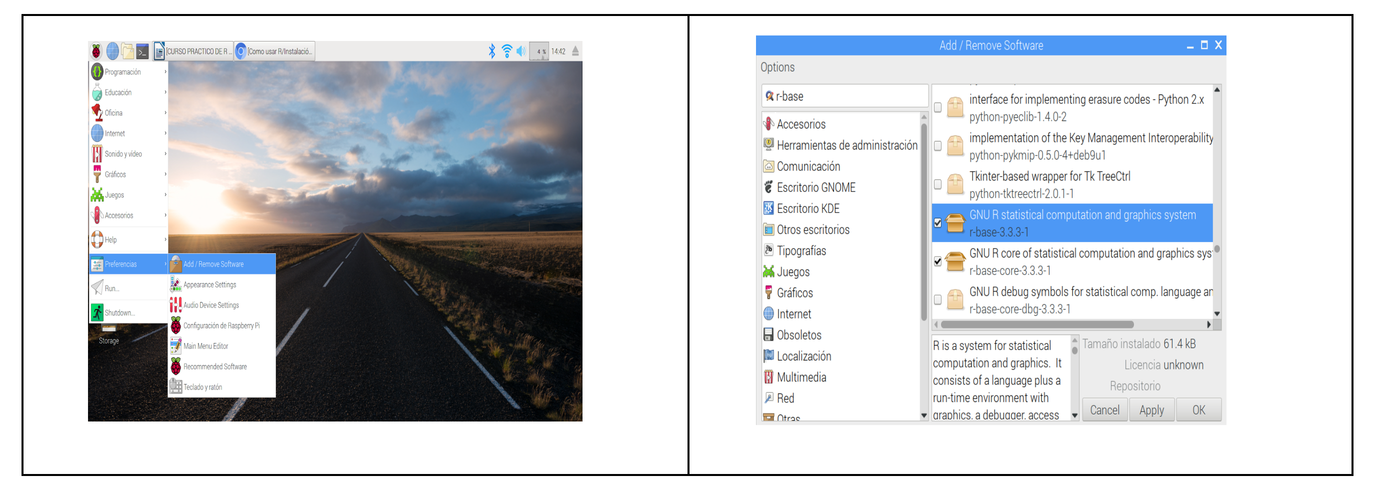
Instalación en Linux para la Raspberry Pi 3

En la mayoría de las distribuciones de Linux el procedimiento es similar. Buscamos en el repositorio de nuestra distribución "r-base" y lo instalamos.

En la Raspberry Pi 3 el sistema operativo es Raspbian una versión adaptada de Debian.

Procedimiento

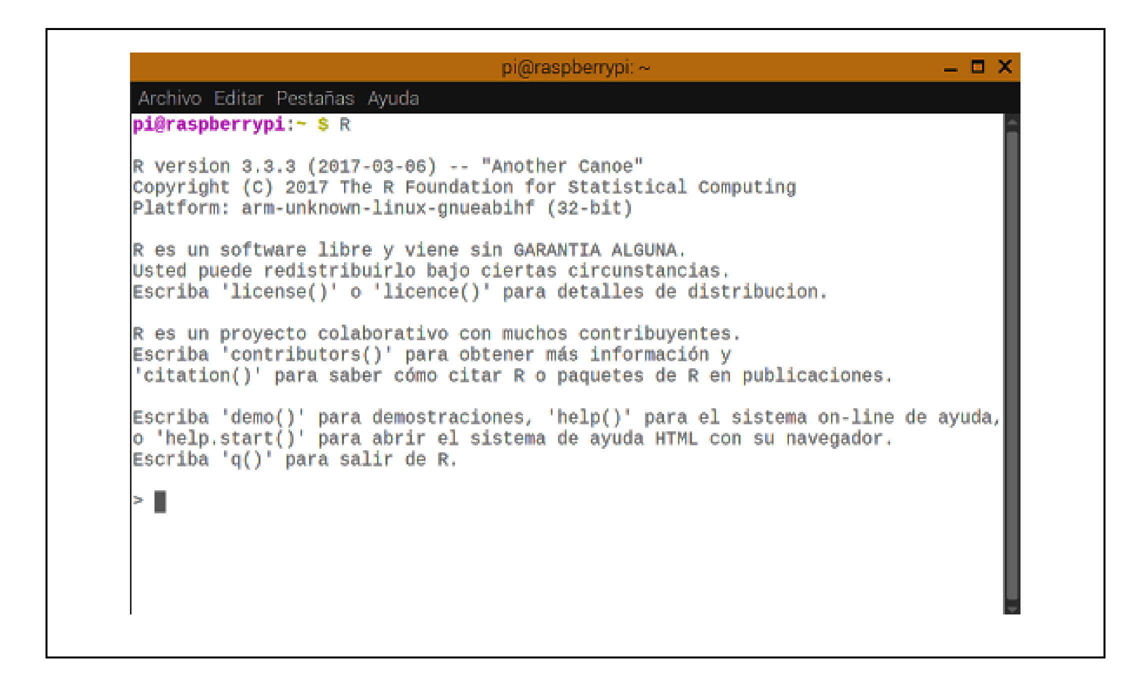
1. Buscamos en el repositorio de nuestra versión el programa R, "r-base" (sin las comillas).



2. Le damos OK y el programa se encarga de bajar el programa y las librerías asociadas.

Para iniciar el programa, entramos a la consola de comandos, escribimos “R” (sin las comillas) y le damos [Enter]

```
pi@raspberrypi: ~
Archivo  Editar  Pestañas  Ayuda
pi@raspberrypi:~ $ R
```



El programa se inicia y nos muestra un “>” que nos indica que está listo para recibir órdenes.



3. Instalación de librerías y directorio de trabajo.

Instalación de Librerías.

En la mayoría de los casos para instalar paquetes desde R es con el comando:

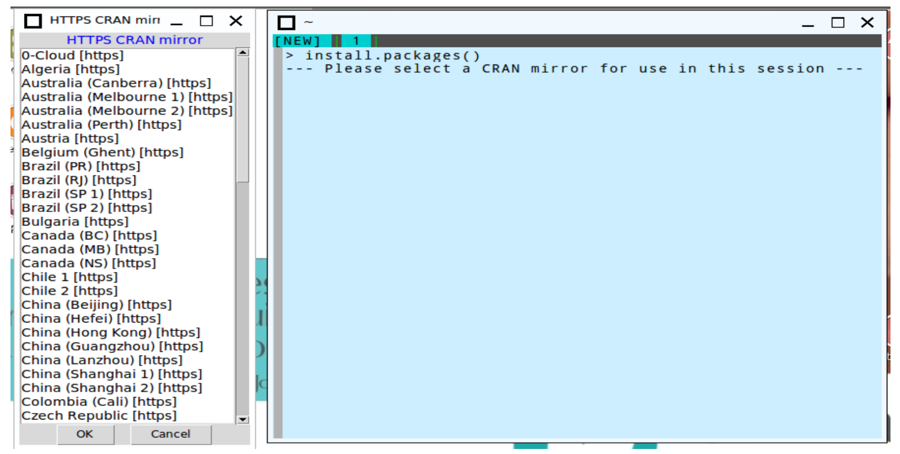
“install.packages()” o “update.packages()”

Colocando en el paréntesis el nombre del paquete que contiene la librería que nos interese. Prácticamente la única librería que se va a utilizar en el presente texto es “fBasics”.

Por lo que para instalarla escribimos desde el R el comando:

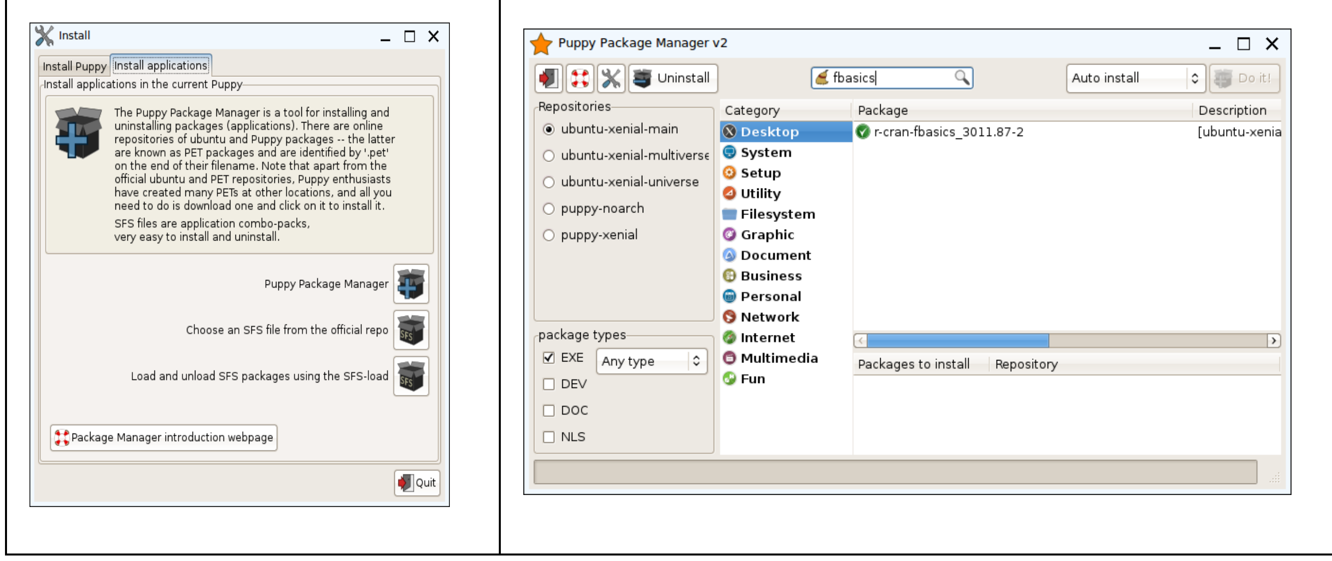
install.packages(“fBasics”)

Nos pedirá que seleccionemos el repositorio a utilizar, puede ser 0-Cloud. Y el programa instala el paquete.



Instalación de librerías desde la pantalla de comandos, directamente desde R.

En el caso del Puppy Linux, no funciona el comando desde R, por lo que debemos buscar en el repositorio del Puppy el paquete que nos interesa. En este caso particular el “fbasics”.



Instalación de librerías desde el repositorio de Puppy linux.

Directorio de datos

El directorio de datos, es en donde vamos a colocar nuestros archivos de datos para poder llamarlos y analizarlos con R. Por omisión R trabaja con el directorio raíz, sin embargo es recomendable modificar esto y tener un archivo dedicado para alojar nuestros datos.

Para determinar el directorio de trabajo por omisión, en el que R va a leer los archivos de datos, utilizamos el comando:

“getwd()”

Para fijar un nuevo directorio de trabajo usamos el comando:

“setwd()”

Por ejemplo:

setwd(“C:/Users/Investigador/Documents/R”) ---- Para Windows

setwd(“/home/data/R”) ---- Para Linux

4. Preparando la base de datos.

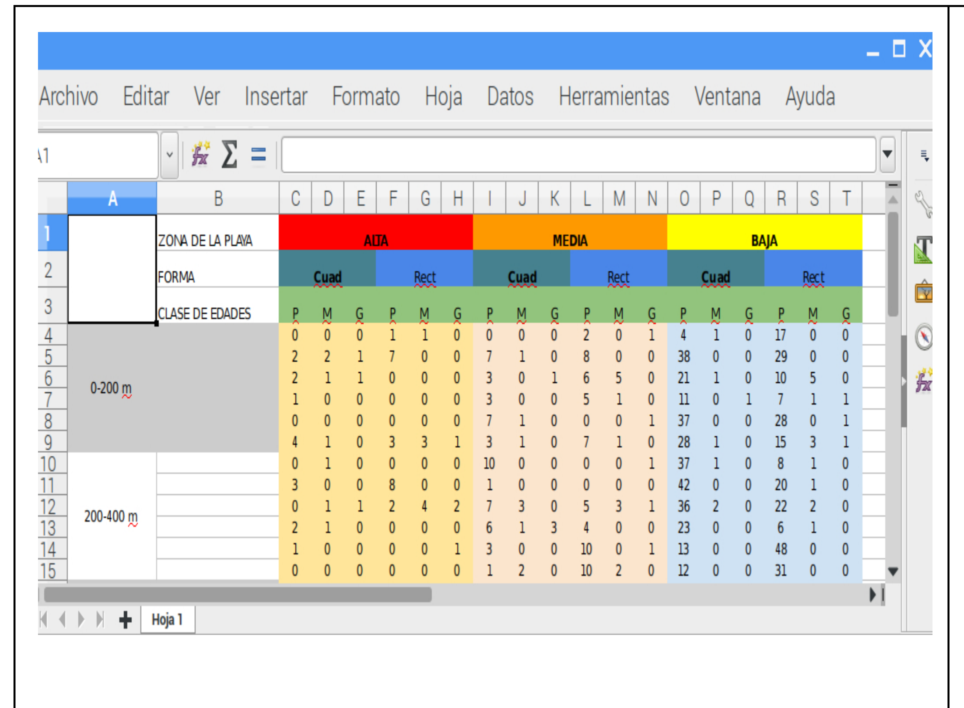
> Una base de datos es un conjunto de datos pertenecientes a un mismo contexto y almacenados sistemáticamente para su posterior uso.
>
> — WikiP

4. Preparando la base de datos.

Uno de los principales problemas que me he encontrado a lo largo de mi carrera docente en estadística e investigación, es la estructura de la base de datos que se desea analizar. En muchos casos es una hoja de cálculo, la cual tiene la peor estructura posible y en algunos casos extremos son tablas elaboradas en un procesador de palabras.

Lo primero que debemos tener en cuenta, es que nuestro archivo de base de datos debe contener el registro de la información y cierta estructura para poder analizarlo de manera efectiva.

> Nuestro archivo de base de datos debe contener el registro de los datos y debe tener cierta estructura específica para poder analizarlo de manera efectiva.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1		ZONA DE LA PLAYA	ALTA						MEDIA						BAJA					
2		FORMA	Cuad			Rect			Cuad			Rect			Cuad			Rect		
3		CLASE DE EDADES	P	M	G	P	M	G	P	M	G	P	M	G	P	M	G	P	M	G
4	0-200 m		0	0	0	1	1	0	0	0	0	2	0	1	4	1	0	17	0	0
5			2	2	1	7	0	0	7	1	0	8	0	0	38	0	0	29	0	0
6			2	1	1	0	0	0	3	0	1	6	5	0	21	1	0	10	5	0
7			1	0	0	0	0	0	3	0	0	5	1	0	11	0	1	7	1	1
8			0	0	0	0	0	0	7	1	0	0	0	1	37	0	0	28	0	1
9			4	1	0	3	3	1	3	1	0	7	1	0	28	1	0	15	3	1
10	200-400 m		0	1	0	0	0	0	10	0	0	0	0	1	37	1	0	8	1	0
11			3	0	0	8	0	0	1	0	0	0	0	0	42	0	0	20	1	0
12			0	1	1	2	4	2	7	3	0	5	3	1	36	2	0	22	2	0
13			2	1	0	0	0	0	6	1	3	4	0	0	23	0	0	6	1	0
14			1	0	0	0	0	1	3	0	0	10	0	1	13	0	0	48	0	0
15			0	0	0	0	0	0	1	2	0	10	2	0	12	0	0	31	0	0

No pueden analizarse datos agrupados o archivos con celdas unidas, textos explicativos, filas o columnas en blanco, etc.

Por lo que revisaremos un poco el cómo preparar de la mejor manera posible, nuestros datos para poder importarlos a R.

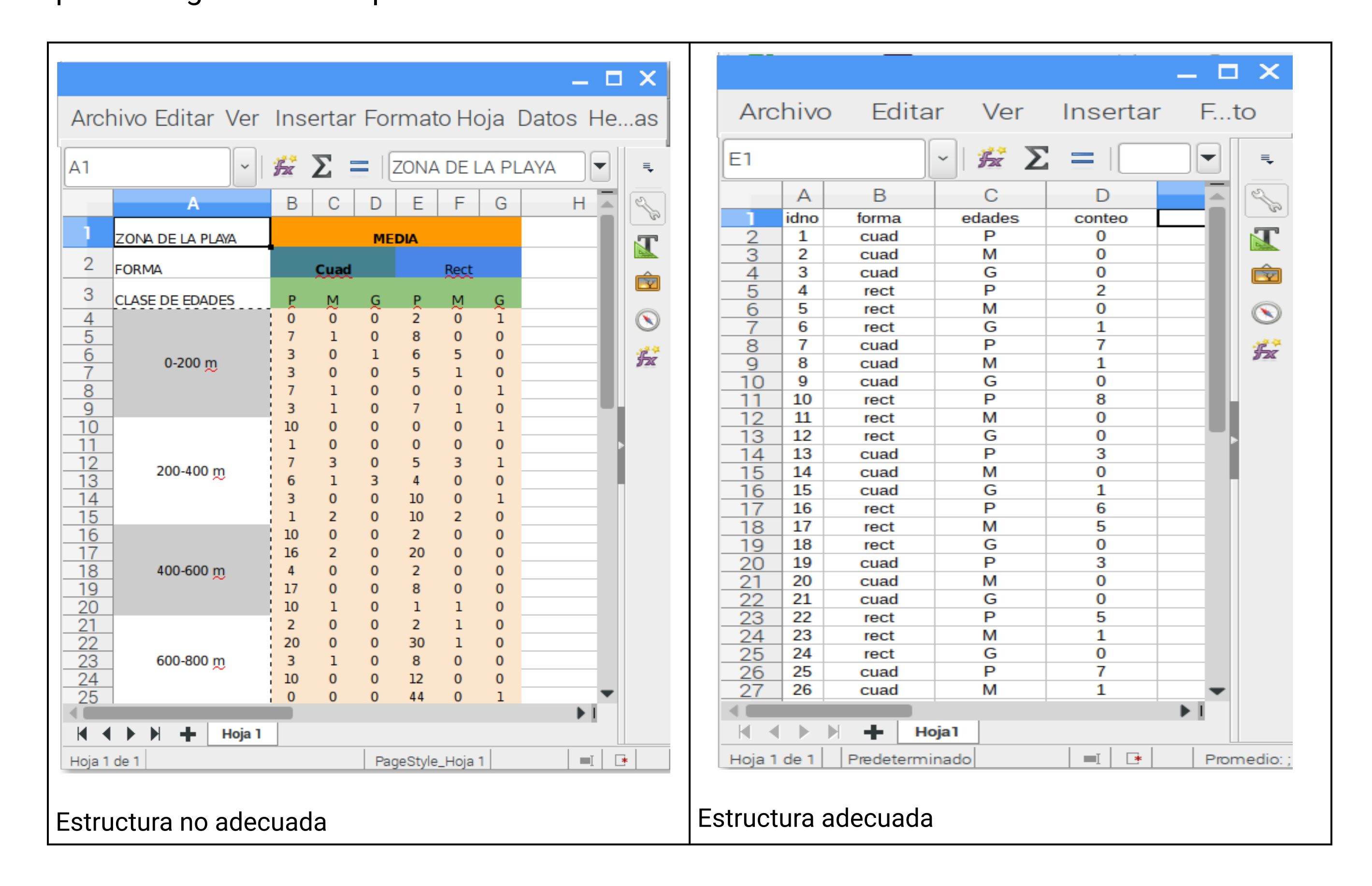
Vamos a asumir que tenemos los datos en una hoja electrónica, puede ser esta Excel, Gnumeric, LibreOffice Calc, o cualquier hoja electrónica que se utilice.

Columnas y Filas

- Las columnas representan a nuestras variables. Los nombres de nuestras variables será el encabezado de las columnas y lo vamos a ubicar en la primera fila.
- Nuestras filas representan nuestras observaciones o unidades de estudio (los sujetos observados). Vamos a emplear la primera columna para ubicar los identificadores de las unidades de estudio. Cada identificador de fila debe ser único, por lo que no debemos tener identificadores repetidos.

Veamos un ejemplo, tomado de un trabajo de investigación de estudiantes de la Facultad de Biología. En este caso se estaba haciendo un recuento de agujeros de cangrejos en una zona de playa a una altura “Media”, se estaban utilizando dos formas geométricas para el recuento de los agujeros de cangrejo, una cuadrada y otra rectangular y se clasificaron los agujeros dependiendo de su tamaño como pequeño (P), mediano (M) o grande (G) como una estimación de la edad del cangrejo.

En la imagen de la izquierda tenemos la hoja electrónica inicial donde se empezaron a vaciar los datos. En la imagen de la derecha tenemos la hoja electrónica ya corregida para poder cargar los datos para su análisis.



	A	B	C	D	E	F	G
1	ZONA DE LA PLAYA	MEDIA					
2	FORMA	Cuad			Rect		
3	CLASE DE EDADES	P	M	G	P	M	G
4	0-200 m	0	0	0	2	0	1
5		7	1	0	8	0	0
6		3	0	1	6	5	0
7		3	0	0	5	1	0
8		7	1	0	0	0	1
9		3	1	0	7	1	0
10	200-400 m	10	0	0	0	0	1
11		1	0	0	0	0	0
12		7	3	0	5	3	1
13		6	1	3	4	0	0
14		3	0	0	10	0	1
15		1	2	0	10	2	0
16	400-600 m	10	0	0	2	0	0
17		16	2	0	20	0	0
18		4	0	0	2	0	0
19		17	0	0	8	0	0
20		10	1	0	1	1	0
21	600-800 m	2	0	0	2	1	0
22		20	0	0	30	1	0
23		3	1	0	8	0	0
24		10	0	0	12	0	0
25		0	0	0	44	0	1

Estructura no adecuada

	A	B	C	D
1	idno	forma	edades	conteo
2	1	cuad	P	0
3	2	cuad	M	0
4	3	cuad	G	0
5	4	rect	P	2
6	5	rect	M	0
7	6	rect	G	1
8	7	cuad	P	7
9	8	cuad	M	1
10	9	cuad	G	0
11	10	rect	P	8
12	11	rect	M	0
13	12	rect	G	0
14	13	cuad	P	3
15	14	cuad	M	0
16	15	cuad	G	1
17	16	rect	P	6
18	17	rect	M	5
19	18	rect	G	0
20	19	cuad	P	3
21	20	cuad	M	0
22	21	cuad	G	0
23	22	rect	P	5
24	23	rect	M	1
25	24	rect	G	0
26	25	cuad	P	7
27	26	cuad	M	1

Estructura adecuada

Los nombres de las columnas (nombres de las variables) deben ser compatibles con las reglas para nombrar variables en R.

Normalmente las hojas iniciales de datos tienen problemas con los nombres, que deben ser corregidos antes de proceder a importar los datos a R.

Corrigiendo la base inicial para hacerla compatible con R.

En el encabezado de las columnas (primera fila) se deben evitar nombres con espacios en blanco. Por ejemplo, si estamos analizando resultados de una competencia de atletismo, en vez de escribir “Salto Largo”, podemos escribir “Salto_largo” o “salto.largo”.

Se deben evitar nombres con símbolos especiales: ?, $, *, +, #, (,), -, /, }, {, |, >, <, etc.

Únicamente el guión bajo y el punto pueden ser usados.

Se debe evitar iniciar el nombre de la variable con un número, por ejemplo para el registro del tiempo en una carrera de 100 metros, debemos evitar “100m” y podemos usar “Tiempo100m” o “T100m”.

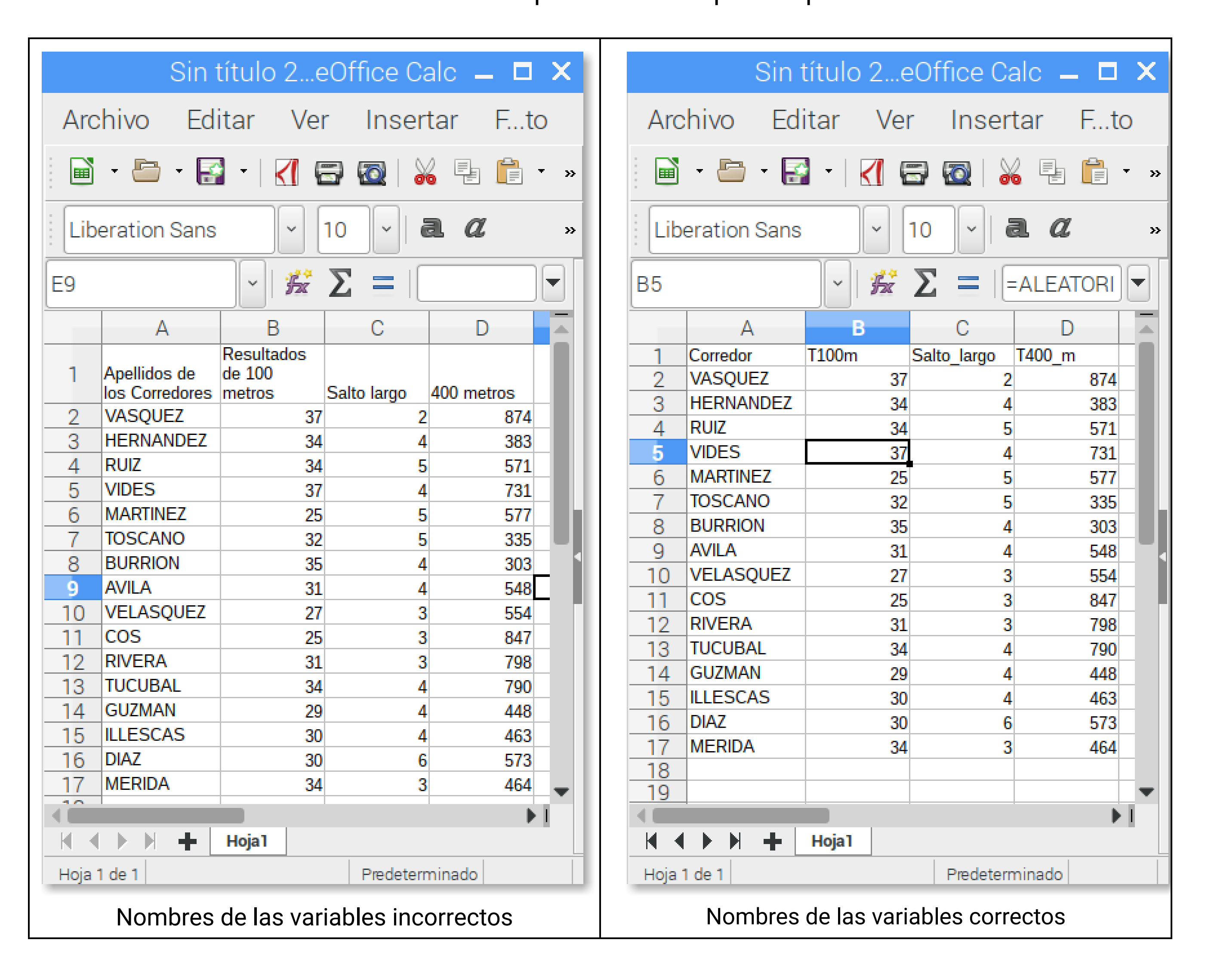
Los nombres de las columnas deben ser únicos, no se permiten nombres duplicados. Es conveniente no poner nombres muy largos, porque complican la sintaxis en el momento del análisis (Generalmente estos nombres corresponden con el nombre de la variable).

Hay que tener presente que:

- R es sensible a las minúsculas y mayúsculas, lo que implica que “tiempo” es distinto a “Tiempo”.
- Se debe evitar dejar filas en blanco.
- Se debe borrar cualquier comentario.
- Se debe reemplazar los valores faltantes por NA (Not Available).
- Para las fechas se debe usar el formato de cuatro dígitos en los años, así se debe usar “01/01/2016” en vez de “01/01/16”.

En las siguientes figuras vemos un ejemplo de nombres incorrectos y nombres correctos para importar los datos a R.

En la figura de la izquierda tenemos nombres de variables incorrectos, mientras que en la de la derecha se muestran nombres compatibles con R para importar los datos.



	A	B	C	D
1	Apellidos de los Corredores	Resultados de 100 metros	Salto largo	400 metros
2	VASQUEZ	37	2	874
3	HERNANDEZ	34	4	383
4	RUIZ	34	5	571
5	VIDES	37	4	731
6	MARTINEZ	25	5	577
7	TOSCANO	32	5	335
8	BURRION	35	4	303
9	AVILA	31	4	548
10	VELASQUEZ	27	3	554
11	COS	25	3	847
12	RIVERA	31	3	798
13	TUCUBAL	34	4	790
14	GUZMAN	29	4	448
15	ILLESCAS	30	4	463
16	DIAZ	30	6	573
17	MERIDA	34	3	464

Nombres de las variables incorrectos

	A	B	C	D
1	Corredor	T100m	Salto_largo	T400_m
2	VASQUEZ	37	2	874
3	HERNANDEZ	34	4	383
4	RUIZ	34	5	571
5	VIDES	37	4	731
6	MARTINEZ	25	5	577
7	TOSCANO	32	5	335
8	BURRION	35	4	303
9	AVILA	31	4	548
10	VELASQUEZ	27	3	554
11	COS	25	3	847
12	RIVERA	31	3	798
13	TUCUBAL	34	4	790
14	GUZMAN	29	4	448
15	ILLESCAS	30	4	463
16	DIAZ	30	6	573
17	MERIDA	34	3	464
18				
19				

Nombres de las variables correctos

5. Importando datos a R.



Importar, dentro del mundo de la informática, este término se refiere a la inclusión de un documento o parte del mismo.

— Tecnologicon

5. Importando datos a R, trabajando desde archivos.

R es capaz de manejar varios formatos de archivos para importar los datos contenidos en nuestra base de datos, para evitar el tener que ingresarlos directamente desde el programa.

Se pueden leer datos desde un archivo nativo de R "archivo.rda", desde un archivo de Excel 97-2003 "archivo.xls", desde un archivo delimitado por comas "archivo.csv".

También está la posibilidad de utilizar paquetes ya hechos para importar datos. Un paquete (package) o librería es una colección de funciones, datos y código R que se almacenan en una carpeta conforme a una estructura bien definida, para que el programa R pueda utilizarlo con facilidad. Son como pequeños programas desarrollados en R para aumentar la capacidad del programa principal. En la web de R se puede consultar la lista de paquetes disponibles. En Enero de 2018 esta lista incluía algo más de 12000 paquetes. Asimismo en la sección Task Views se puede consultar una lista de paquetes ordenada según áreas de aplicación.

Cuando instalamos R se incorporan por defecto numerosas librerías. Podemos ver una lista de las librerías que actualmente tenemos instalados en nuestro ordenador ejecutando:

```
>library()
```

Dentro de estos paquetes o librerías, existen algunas que nos permiten importar datos desde otros programas como el SPSS “archivo.sav”, sin embargo como uno de los objetivos de este texto es hacer las cosas prácticas sin la necesidad de estar descargando librerías, usaremos el enfoque de exportar los datos desde el programa en que los tengamos a un archivo delimitado por comas “archivo.csv”, la mayoría de los programas pueden hacer esto y ya con el archivo csv , simplemente lo cargamos a R.

Importando datos desde un archivo csv.

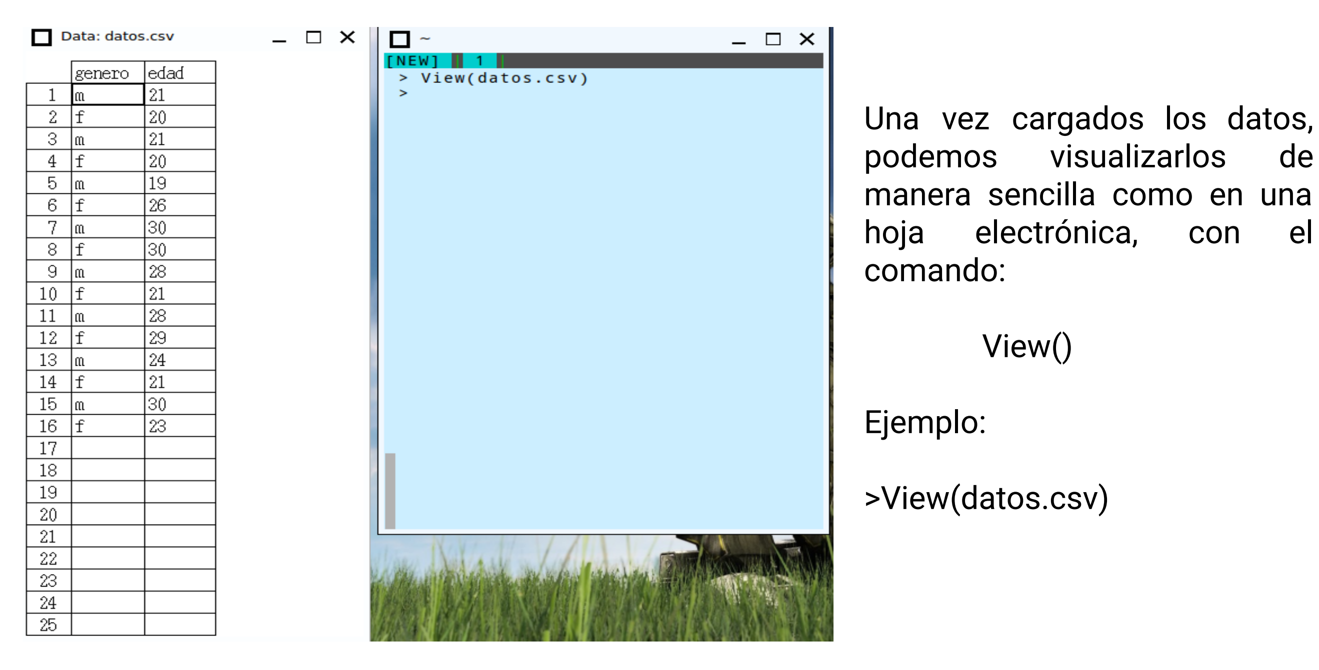
Los archivos CSV (del inglés comma-separated values) son un tipo de documento en formato abierto sencillo que se utilizan para representar datos en forma de tabla, en las que las columnas se separan por comas y las filas por saltos de línea.

La mayoría de programas pueden guardar los datos en un formato delimitado por coma o csv y en R podemos leer directamente los datos de un archivo csv, sin la necesidad de ninguna librería.

Lo hacemos con la función “read.csv()”

Ejemplo:

>datos.csv=read.csv(“nombrearchivo.csv”, header=T)



	genero	edad
1	m	21
2	f	20
3	m	21
4	f	20
5	m	19
6	f	26
7	m	30
8	f	30
9	m	28
10	f	21
11	m	28
12	f	29
13	m	24
14	f	21
15	m	30
16	f	23
17		
18		
19		
20		
21		
22		
23		
24		
25		

Una vez cargados los datos, podemos visualizarlos de manera sencilla como en una hoja electrónica, con el comando:

View()

Ejemplo:

>View(datos.csv)

Visualización de los datos con el comando View()

Importando datos desde un archivo de Excel

5.1. La manera más sencilla.

La manera más sencilla de importar datos desde Excel es guardar los datos desde Excel en formato csv delimitado por comas y luego importarlo como archivo csv desde R.

5.2. Importar directamente de Excel, librería “gdata”.

Es posible importar a R directamente desde un archivo de Excel, pero para esto es necesario el apoyo de la librería “gdata” que debe ser descargada previamente a nuestro programa de R.

Para cargar la librería “gdata” en R debemos escribir los siguientes comandos:

- install.packages(“gdata”)

- library(gdata)

Luego utilizamos el comando:

datos<-read.xls(“nombrearchivo.xls”, sheet=1)

Hay que tomar en cuenta que sólo lee formato “xls”, es decir Excel 97-2003.

5.3. Importar a través de Copiar y Pegar.

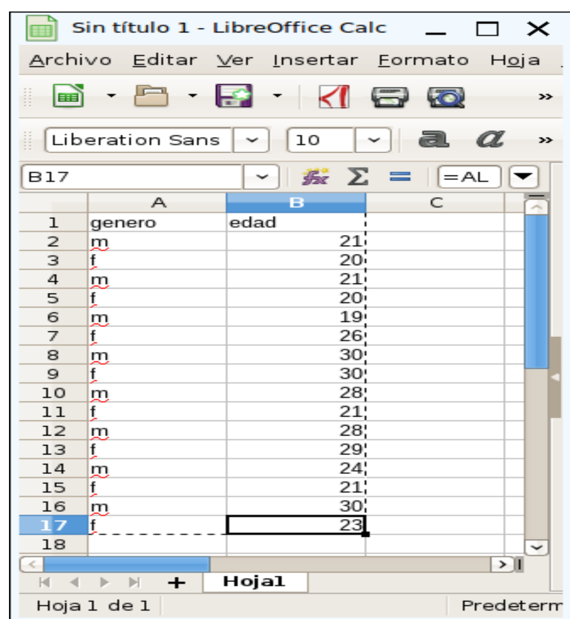
Si no se desea instalar librerías extra, podemos también copiar los datos a través de copiar y pegar (copy / paste).

Esto se hace marcando los datos en Excel y luego los copiamos con [CTRL]+[C] .

Por ejemplo copiamos los datos del género y edad de un archivo de Excel o cualquier otra hoja electrónica.

Luego cargamos los datos a un data.frame para poder analizarlos en R, escribimos:

nombre=read.delim(“clipboard”)



	A	B	C
1	genero	edad	
2	m	21	
3	f	20	
4	m	21	
5	f	20	
6	m	19	
7	f	26	
8	m	30	
9	f	30	
10	m	28	
11	f	21	
12	m	28	
13	f	29	
14	m	24	
15	f	21	
16	m	30	
17	f	23	
18			

Ejemplo del comando para crear el data.frame llamado datos3 `>datos3=read.delim("clipboard")` Luego llamamos a datos3 para ver el contenido del data.frame `>datos3`	Al llamar a datos3, el programa nos muestra el contenido del data.frame `   género   edad` `1    m       21` `2    f       20` `3    m       21` `4    f       20` `5    m       19` `6    f       26` `    etc.`

Si deseamos calcular estadísticas con los datos recién ingresados, debemos extraerlos a un vector, así:

Para extraer los datos de edad a un vector llamado datos.edad, >datos.edad=c(datos3$edad) >datos.edad Así podemos, por ejemplo, calcular la media aritmética; >mean(datos.edad)	(respuestas de R a los comandos) [1] 21 20 21 20 19 26 30 30 28 21 28 29 24 21 30 23 [1] 24.4375

Importando datos desde un archivo TXT.

Para importar datos desde un archivo TXT, un archivo de texto separado por comas, punto y coma o por tabulaciones, podemos usar el comando:

"read.delim()" o "read.table()"

Lo importante es usar el argumento "sep" que indica el separador empleado.

En el caso que tengamos el archivo "datos.txt" con los datos separados por tabulaciones, empleamos:

datos=read.delim("datos.txt", header=TRUE, sep="\t")

Para los distintos separadores usamos:

Para tabulaciones empleamos: "\t"

Para comas empleamos: ","

Para punto y coma empleamos: ";"

Etc.



6. Archivos de salidas.

Guardando Gráficas

Para guardar las gráficas producidas en R, podemos utilizar varios formatos, dentro de los que están: como imagen jpg o como archivo pdf.

Archivo gráfico jpg . jpg()

Teniendo dos series de datos “x” y “y”:

```
> x=rnorm(50,40,10)
> y=rnorm(70,50,5)
```

* puede consultar la sección Generación de datos, pag.38

- Primero especificamos el formato de gráfico jpg con el comando “jpg()” .
Dentro de los paréntesis especificamos el nombre del archivo, en este caso “Graf01.jpg”

- Luego damos el comando de gráfica, en este caso una gráfica de caja y bigotes para ambas series.

- Por último cerramos el dispositivo gráfico, a través del comando “dev.off()”

> jpeg(filename="Graf01.jpg") > boxplot(x,y) > dev.off() >null device	

```
Algunos argumentos para modificar el gráfico.

jpeg(filename="Grafico3.jpeg"), #Nombre y extensión
   width = 33,  #Anchura
   height= 19,  #Altura
   res = 72  ,  #Resolución 72 ppi es un estándar
   units ="cm", #Unidades
plot(df$x,df$y) #Gráfico
dev.off()       #Cierre del archivo
```

Archivo PDF, pdf().

Podemos guardar nuestro gráfico como un archivo PDF, a través del comando pdf(), de la misma manera que se guarda un gráfico jpg.

Para los mismos datos “x” y “y” trabajados anteriormente.

pdf(file="Graf02.pdf") boxplot(x,y) dev.off()	>null device

Archivo de salidas “archivo.txt” y archivo nativo de R, “archivo.rda”

R genera diversos tipos de resultados: datos, tablas, gráficos, etc. existen varias opciones para guardar estos resultados dependiendo de lo que queremos realizar después con los resultados.

6.1. Para guardar información NO gráfica en un fichero de texto.

En ocasiones sólo queremos salvar los resultados no gráficos, que produce R y que generalmente observamos en la pantalla.

La manera más sencilla es con “copiar y pegar”.

Marcamos los resultados que nos interesan con el mouse.

Si estamos trabajando en windows los copiamos con [CTRL] + [C] y luego los pegamos en nuestro procesador de textos con [CTRL]+[V]

En Linux para copiar desde la consola de comandos no se utiliza [CTRL] + [C].

Al marcar el texto automáticamente se copia en la memoria, así que después de marcar el texto, nos ubicamos en nuestro procesador de textos y oprimimos el botón central en el mouse (en la mayoría de los mouse modernos es la rueda de desplazamiento entre los dos botones) esto es el equivalente del [CTRL]+[V] y los resultados que nos interesan se pegan en nuestro procesador de textos.

Otra forma es redireccionar la salida de la pantalla a un archivo de texto.

En ese caso utilizaremos las función sink , especificando un archivo de texto donde se guardarán los resultados obtenidos.

Primero comprobamos cuál es el directorio de trabajo (working directory o wd):

getwd()	> [1] "/home"

Es posible cambiar el directorio de trabajo (working directory o wd) con la función setwd()

setwd(“home/R”) getwd()	> [1] "/home/R"

Con la función sink hacemos que la salida de R sea el fichero de texto que indiquemos:

sink("resultados.txt") a <- c(1, 2, 3, 5) summary(a) print(a)	

Si ya hemos terminado de guardar resultados, redirigimos la salida a la pantalla (como estaba al principio).

sink() print(a)	> [1] 1 2 3 5

En el archivo “resultados.txt” tendremos:

```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   1.00    1.75    2.50    2.75    3.50    5.00 

[1] 1 2 3 5
```

Al utilizar este método lamentablemente no tenemos retroalimentación con respecto a los datos que se están guardando en el momento de estar trabajando en R, por lo que este método podría ser más útil dentro de un programa o proceso automatizado.

6.2. Guardando una base de datos (data.frame o matrix)

Tras generar una base de datos (data.frame o matrix) es posible guardar ese objeto específico para que podamos utilizarlo en sesiones subsecuentes o para archivarlo como referencia.

Esto podemos hacerlo con varias funciones como write.table, write.csv o write.matrix:

Así, para los datos:

talla <- c(150, 156, 166, 175, 175) peso <- c(75, 70, 72, 80, 55) provincia <- c("Alicante", "Castellon", "Valencia", "Valencia", "Alicante") datos <- data.frame(talla, peso, provincia) datos *#nos da el contenido del data.frame*	> talla peso provincia > 1 150 75 Alicante > 2 156 70 Castellon > 3 166 72 Valencia > 4 175 80 Valencia > 5 175 55 Alicante

Para guardar este data.frame en un archivo de texto, escribimos:

```
write.table(datos, "misdatos.txt", sep = "\t", quote = F, row.names = F)
```

6.3. Guardar los objetos generados en la sesión de R en un fichero .rda

Con las funciones save y save.image es posible guardar todos o algunos objetos generados en la sesión de R.

Esto nos permite guardar nuestro trabajo, matrices y/o vectores para futuros análisis o cuando el análisis se hace en varias sesiones.

Para volver a cargar la información en una sesión subsecuente utilizamos la función load().

Por ejemplo, si tenemos:

talla <- c(150, 156, 166, 175, 175) peso <- c(75, 70, 72, 80, 55) provincia <- c("Alicante", "Castellon", "Valencia", "Valencia", "Alicante") datos <- data.frame(talla, peso, provincia) datos # *visualizamos el contenido de "datos"*	> talla peso provincia > 1 150 75 Alicante > 2 156 70 Castellon > 3 166 72 Valencia > 4 175 80 Valencia > 5 175 55 Alicante

Y tenemos también:

datos2 <- datos[, 1:2] head(datos2) # Visualiza las primeras 6 líneas	#> talla peso #> 1 150 75 #> 2 156 70 #> 3 166 72 #> 4 175 80 #> 5 175 55

Verificamos que objetos tenemos en la memoria con el comando ls()

ls()	>[1]"datos" "datos2" "peso" "provincia" “talla"

Para guardar todos los objetos en un único fichero específico de R, escribimos: save.image(file = "resultados.rda")

Para cargar todos los objetos en una sesión posterior de R, escribimos: load("resultados.rda")

Demostración.

Inmediatamente después de crear el archivo “resultados.rda” cerramos la sesión de R, con q() y cerramos la ventana de comandos.	> q() Save workspace image? [y/n/c]: n
Reiniciamos R, desde la ventana de comandos.	pi@usuario:~ $ R
Verificamos que la memoria no contiene objetos.	> ls() character(0)
Cargamos el archivo guardado anteriormente,	> load("resultados.rda")
Al verificar la memoria vemos que ahora están disponibles los objetos que habíamos guardado anteriormente.	> ls() [1] "datos" "datos2" "peso" "provincia" "talla"

"Si sólo queremos guardar alguno o algunos objetos, se puede salvar con la función save:

save(datos, datos2, file = "resultados2.rda")





> Un dato es una representación simbólica (numérica, alfabética, algorítmica, espacial, etc.) de un atributo o variable cuantitativa o cualitativa.
>
> — WikiP

7. Generación de datos y datos desde el teclado.

Para generar una serie de números aleatorios con distribución normal.

Una función muy útil cuando estamos utilizando datos para ejercicios o para practicar el análisis de los mismos, es la generación de números aleatorios que presenten un comportamiento normal.

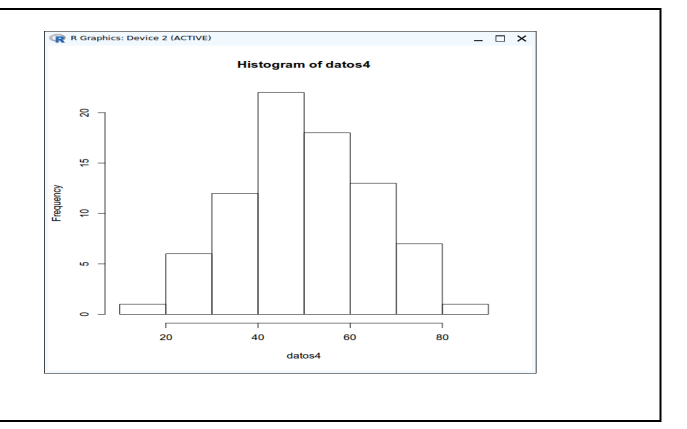
Esto lo hacemos con el comando:

rnorm(n, mean=0, sd=1)

- donde "n" va a ser el número de datos a generar.
- mean= aquí colocamos la media aritmética de la serie de datos.
- sd= aquí colocamos la desviación estándar de la serie de datos.

Ejemplo: Para generar una serie de 80 datos con una media de 50 y una desviación estándar de 15, escribimos:

```
datos4=rnorm(80,mean=50, sd=15)
```

datos4=rnorm(80,mean=50, sd=15) También podemos escribir: datos4=rnorm(80,50,15)	

Podemos revisar el comportamiento general del grupo con un histograma o podemos revisar los datos directamente a través del comando “View(datos4)” .

Para un número moderado de datos puede resultar más útil el empleo de un diagrama de tallo y hojas con:

stem(datos4)

stem(datos4)	The decimal point is 1 digit(s) to the right of the | 1 | 2 | 0225688 3 | 333445566779 4 | 0011223345577788889999 5 | 011112234567778899 6 | 0111233567799 7 | 01123679 8 | 3

Para generar una serie de números aleatorios con distribución uniforme.

Para generar una distribución uniforme de datos utilizamos: runif()

La sintaxis es:

runif(“n datos”, “Li”, “Ls”)

Donde: “n datos” = número de datos a generar.

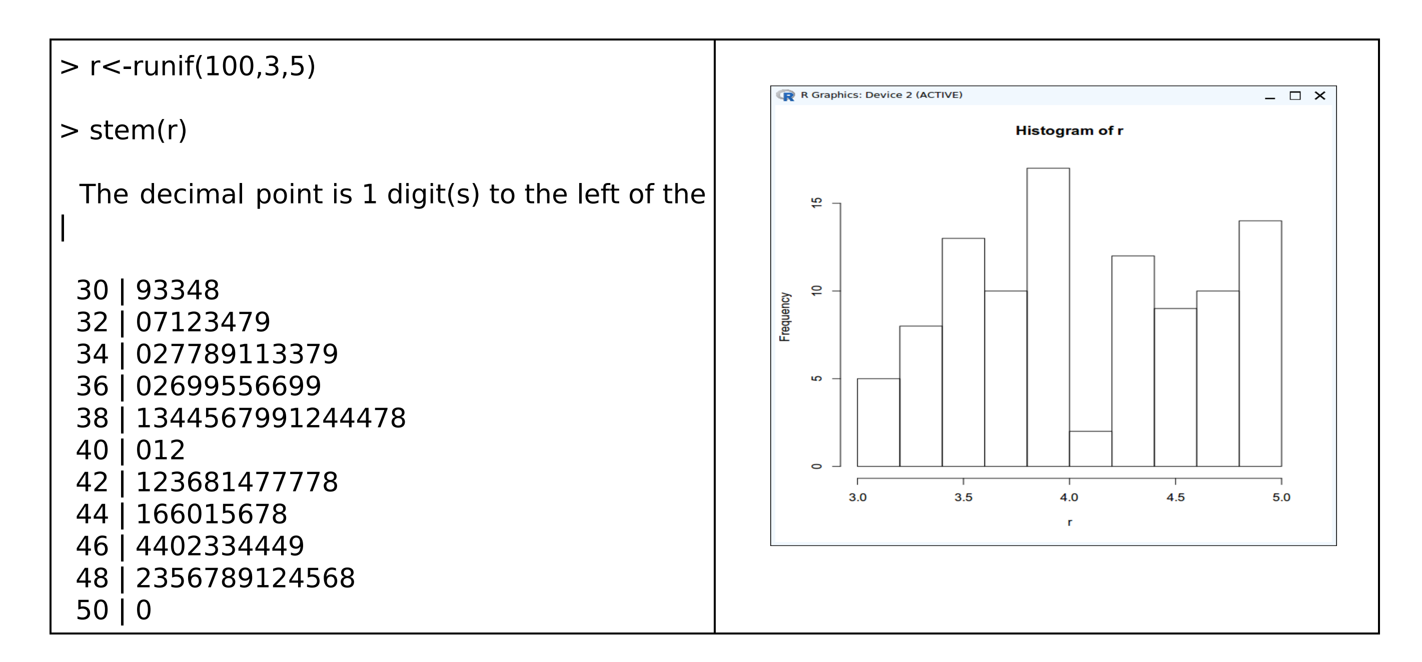
“Li” = Límite inferior de la serie de datos.

“Ls” = Límite superior de la serie de datos.

Ejemplo:

Para generar 100 datos de distribución uniforme contenidos entre 3 y 5 escribimos:

```
r<-runif(100,3,5)
```



```
> r<-runif(100,3,5)

> stem(r)

  The decimal point is 1 digit(s) to the left of the
|

  30 | 93348
  32 | 07123479
  34 | 027789113379
  36 | 02699556699
  38 | 1344567991244478
  40 | 012
  42 | 123681477778
  44 | 166015678
  46 | 4402334449
  48 | 2356789124568
  50 | 0
```


Para generar una serie de números aleatorios con distribución exponencial.

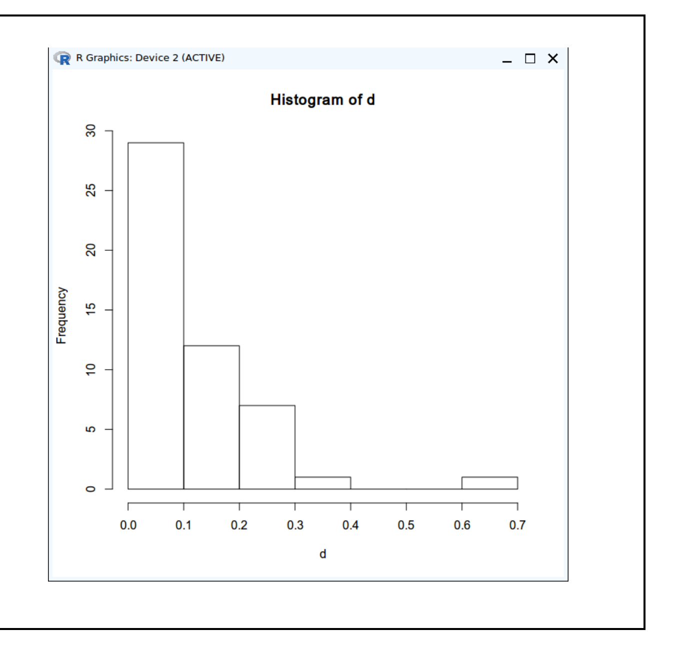
Para generar una distribución exponencial de datos utilizamos: rexp()

La sintaxis es:

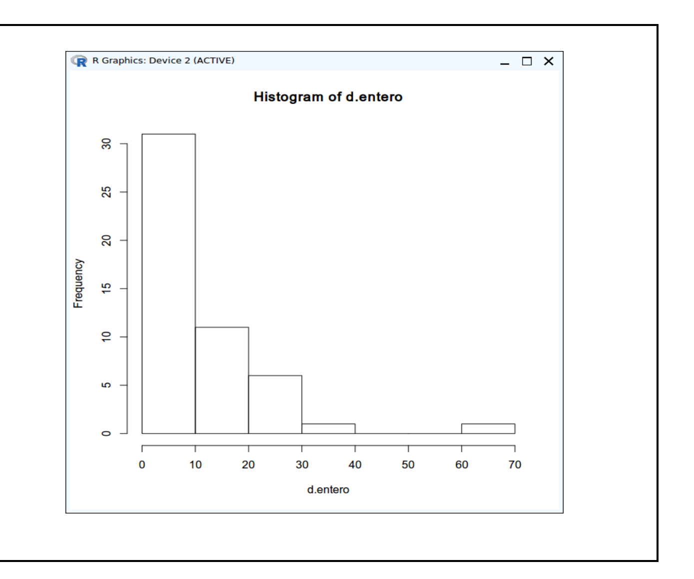
rexp(“n datos”,“Ls”)

Donde:	“n datos”	= número de datos a generar.
	“Ls”	= Límite superior de la serie de datos.

Ejemplo: Para generar 50 datos de distribución exponencial contenidos entre 0 y 0.8 escribimos:

> d=rexp(50,8) > stem(d) The decimal point is 1 digit(s) to the left of the | 0 | 11122222334444455555556778888 1 | 0011145588999 2 | 0444666 3 | 4 4 | 5 | 6 | 9	

Podemos transformar los resultados a través de operaciones aritméticas, así por ejemplo si deseamos trabajar con números enteros, multiplicamos los resultados contenidos en “d” por 100 y eliminamos los decimales a través de la función “as.integer()”

> d.entero=as.integer(d*100) > head(d.entero) [1] 1 6 4 0 4 18	

Ingresando datos desde el teclado.

7.1. A través de la creación de un vector.

Para pocos datos podemos ingresar un vector de datos, escribiendo directamente los datos separados por coma.

Ejemplo:

```
a=c(1,2,3,4,5,6,7,8,9,10)
```

7.2. A través del comando scan()

Una manera más sencilla, cuando se maneja un número medio de datos, es a través del comando scan(), el cual nos permite ingresar los datos de uno en uno, directamente desde el teclado numérico, oprimiendo [Enter] después de cada número.

Los datos dejan de cargarse al presionar [Enter] sin haber digitado ningún dato.

Ejemplo:

>b=c(scan()) 1: 20 2: 21 3: 22 4: 23 5: 24 6: 25 7: Read 6 items >b #visualiza el contenido de b	[1] 20 21 22 23 24 25 >

7.3. Cargando datos agrupados

Cuando los datos de un ejercicio están presentados a través de datos agrupados y para no introducir individualmente cada uno de los datos podemos usar el modificador “rep”.

Ejemplo: Para cargar los datos del cuadro

```
--------------------------------
Intervalos      x        f
--------------------------------
4 – 4.9        4.45       3
5 – 5.9        5.45     102
6 – 6.9        6.45     225
7 – 7.9        7.45     219
8 – 8.9        8.45     100
9 – 9.9        9.45       1
--------------------------------
                         650
--------------------------------
```

Usamos la marca de clase “x” y la frecuencia, colocando el modificador “rep” frente a los paréntesis donde ubicamos la marca de clase “,” y la frecuencia;

```
rep(“marca de clase”, “frecuencia”)
```

cuadro=c(rep(4.45,3),rep(5.45,102),rep(6.45,225),rep(7.45,219),rep(8.45,100),rep(9.45,1))

De esta forma cargamos 650 datos y podemos analizarlo de manera sencilla. Un resumen del cuadro lo podemos obtener con el comando “table()”

table(cuadro)

> table(cuadro) cuadro	4.45 5.45 6.45 7.45 8.45 9.45 3 102 225 219 100 1

Un resumen de los datos lo obtenemos con el comando:

```
summary()
```

> summary(cuadro)	Min. 1st Qu. Median Mean 3rd Qu. Max. 4.450 6.450 6.450 6.933 7.450 9.450

Hay que tomar en cuenta que al utilizar las marcas de clase y no los verdaderos datos la mediana es un resultado aproximado y es necesario calcularla de la manera tradicional con los datos del cuadro y la fórmula:

$$Md=Li + (f/j)\ (Ui - Li)$$

Cargando datos agrupados usando dos vectores

Otra manera más eficiente de cargar el cuadro, es a través de dos vectores, uno con las marcas de clase y el otro con las frecuencias, de la siguiente manera:

```
>marca=c(4.45,5.45,6.45,7.45,8.45,9.45)
>frec=c(3,102,225,219,100,1)
```

Luego lo integramos en un vector llamado cuadro1

>cuadro1=c(rep(marca,frec)) > table(cuadro1) #nos muestra un cuadro con las marcas de clase y sus frecuencias	cuadro1 4.45 5.45 6.45 7.45 8.45 9.45 3 102 225 219 100 1 >





Los data frames son estructuras de datos de dos dimensiones (rectangulares) que pueden contener datos de diferentes tipos, por lo tanto, son heterogéneas.

— R

8. Trabajando con data.frames .

Explorando un data.frame

Una vez importados los datos a un data.frame, como por ejemplo en “iris” que se incluye en el paquete base de R, podemos revisar su contenido de varias formas, y podemos extraer los valores de una variable en particular para el análisis de los datos.

8.1. View()

Nos permite visualizar los datos en una ventana, similar a una hoja electrónica.

Esto nos permite determinar el nombre de las variables, el tipo de las mismas, etc.

Data: iris

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa
9	4.4	2.9	1.4	0.2	setosa
10	4.9	3.1	1.5	0.1	setosa
11	5.4	3.7	1.5	0.2	setosa
12	4.8	3.4	1.6	0.2	setosa
13	4.8	3.0	1.4	0.1	setosa
14	4.3	3.0	1.1	0.1	setosa
15	5.8	4.0	1.2	0.2	setosa
16	5.7	4.4	1.5	0.4	setosa
17	5.4	3.9	1.3	0.4	setosa
18	5.1	3.5	1.4	0.3	setosa
19	5.7	3.8	1.7	0.3	setosa
20	5.1	3.8	1.5	0.3	setosa
21	5.4	3.4	1.7	0.2	setosa
22	5.1	3.7	1.5	0.4	setosa
23	4.6	3.6	1.0	0.2	setosa
24	5.1	3.3	1.7	0.5	setosa
25	4.8	3.4	1.9	0.2	setosa

```
[NEW] 1
> View(iris)
```

8.2. head()

Nos presenta las primeras 6 filas de la base de datos.

```
> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

8.3. tail()

Nos presenta las 6 últimas filas de la base de datos.

```
> tail(iris)
    Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
145          6.7         3.3          5.7         2.5 virginica
146          6.7         3.0          5.2         2.3 virginica
147          6.3         2.5          5.0         1.9 virginica
148          6.5         3.0          5.2         2.0 virginica
149          6.2         3.4          5.4         2.3 virginica
150          5.9         3.0          5.1         1.8 virginica
```

8.4. names()

Nos da los nombres de las variables.

```
> names(iris)
[1]   "Sepal.Length"   "Sepal.Width"      "Petal.Length"   "Petal.Width"
"Species"
```

Extracción de una variable conociendo el nombre. (Extracción de un vector)

Para extraer la longitud del pétalo a un vector llamado “longitud.petalo”, debemos saber que el nombre de la variable es “Petal.Width”.
El comando es:

“vector” = “base”$”nombre de la variable”

longitud.petalo=iris$Petal.Width

> summary(longitud.petalo)	<pre>Min. 1st Qu. Median Mean 3rd Qu. Max. 0.100 0.300 1.300 1.199 1.800 2.500</pre>

Si descargamos el paquete “fBasics” y lo cargamos a la memoria.

```
>install.packages(fBasics)

>library(fBasics)
```

Tenemos el comando “basicStats” que nos da un resumen completo con todas las estadísticas descriptivas de uso general en un solo comando.

Así:

```
>basicStats(longitud.petalo)
```

```
            longitud.petalo
nobs             150.000000
NAs                0.000000
Minimum            0.100000
Maximum            2.500000
1. Quartile        0.300000
3. Quartile        1.800000
Mean               1.199333
Median             1.300000
Sum              179.900000
SE Mean            0.062236
LCL Mean           1.076353
UCL Mean           1.322313
Variance           0.581006
Stdev              0.762238
Skewness          -0.100917
Kurtosis          -1.358179
```

Combinación de dos vectores en un data.frame

En algunos casos, es preferible combinar dos vectores en un data.frame para facilitar la aplicación de algunas pruebas estadísticas.

Para combinar los vectores "peso" y "long" en un data.frame llamado dat1 lo hacemos a través del comando:

```
>dat1 = data.frame(peso,long)
```

Extracción de un vector de un data.frame en base a uno o más factores.

En muchas ocasiones, nos interesa el análisis de un vector en base a un factor, como por ejemplo "el análisis de las calificaciones de los alumnos en base a un curso en particular" o "resultado de concentración de calcio en sangre en base a un medicamento para la osteoporosis" o "la longitud del fémur dependiendo de una especie particular de ratones".

Para poder explorar las maneras en que podemos extraer estos vectores en particular, debemos preparar previamente un data.frame para que nos sirva de ejemplo.

Elaborando un data.frame de ejemplo.

Vamos a elaborar un data.frame llamado "Datos", con la combinación de dos vectores:

- Un vector llamado "grupo", con 30 datos de cada uno de los grupos "A", "B", "C".
- Un vector llamado "valor", con 90 valores aleatorios con distribución normal con una media de 60 y desviación estándar de 10, asi:

```
> grupo=c(rep("A",30),rep("B",30),rep("C",30))
> valor=rnorm(90,mean=60,sd=10)
> Datos=data.frame(grupo,valor)
```

Haciendo una exploración del data.frame "Datos", tenemos:

> head(Datos)	<pre> grupo valor 1 A 51.24227 2 A 56.02579 3 A 65.92976 4 A 44.12616 5 A 47.26777 6 A 65.40705</pre>

Podemos obtener estadísticas de cada grupo a través del comando tapply()

>tapply(datos$valor,datos$grupo,mean)	<pre> A B C 58.75425 59.11764 59.80312</pre>
> tapply(datos$valor,datos$grupo,sd)	<pre> A B C 4.874735 4.178205 5.017734</pre>

Podemos ver una descripción estadística general del grupo con el comando summary()

> summary(Datos)	<pre>grupo valor A:30 Min. :44.13 B:30 1st Qu.:54.98 C:30 Median :61.83 Mean :61.90 3rd Qu.:68.54 Max. :83.09</pre>

Ahora bien, si nos interesa un análisis más completo del comportamiento individual de cada grupo (A, B y C), la manera más sencilla de obtenerlo es extrayendo un vector por cada grupo.

Extracción de un vector dependiendo del valor de un factor.

Con nuestro data.frame de ejemplo creado en la sección anterior, deseamos generar todas las estadísticas descriptivas de la variable "valor" para el grupo A, el grupo B y el grupo C de manera individual.

Para hacer esto extraemos en un vector los valores de la variable “valor” dependiendo del valor de la variable “grupo”, de la siguiente manera:

```
> grupo.a=subset(Datos$valor,Datos$grupo=="A")
> grupo.b=subset(Datos$valor,Datos$grupo=="B")
> grupo.c=subset(Datos$valor,Datos$grupo=="C")
```

Esto crea un vector llamado grupo.a, con los datos de la variable “valor” cuando el grupo tiene un valor de A. Un vector llamado grupo.b con los datos de la variable “valor” cuando el grupo tiene un valor de B y un vector llamado grupo.c con los datos de la variable “valor” cuando el grupo tiene un valor de C.

Podemos correr todas las estadísticas descriptivas para cada vector con el comando basicStats(), de la librería fBasics. (Ver “Instalación de librerías”, pág.19 y “Descripción de grupos. Estadística descriptiva, basicStats.”, pág.59)

Imaginemos que ya hemos instalado el paquete fBasics. (esto se hace con install.packages(fBasics), pág 19)

Cargamos la librería a memoria a través de:

library(fBasics)

y luego corremos el comando basicStats() para cada vector recién creado, así:

```
> basicStats(grupo.a)
                  grupo.a
nobs            30.000000
NAs              0.000000
Minimum         47.379598
Maximum         70.658958
1. Quartile     56.653246
3. Quartile     61.205649
Mean            58.754247
Median          58.609270
Sum           1762.627423
SE Mean          0.890001
LCL Mean        56.933992
UCL Mean        60.574503
Variance        23.763040
Stdev            4.874735
Skewness        -0.158603
Kurtosis         0.453643
```

```
> basicStats(grupo.b)
                  grupo.b
nobs            30.000000
NAs              0.000000
Minimum         48.516922
Maximum         66.596123
1. Quartile     56.435376
3. Quartile     60.974341
Mean            59.117644
Median          59.362263
Sum           1773.529325
SE Mean          0.762832
LCL Mean        57.557477
UCL Mean        60.677811
Variance        17.457394
Stdev            4.178205
Skewness        -0.376071
Kurtosis        -0.047607
```

```
> basicStats(grupo.c)
                  grupo.c
nobs            30.000000
NAs              0.000000
Minimum         49.947641
Maximum         68.980376
1. Quartile     56.924631
3. Quartile     63.858262
Mean            59.803119
Median          59.537962
Sum           1794.093582
SE Mean          0.916109
LCL Mean        57.929467
UCL Mean        61.676772
Variance        25.177657
Stdev            5.017734
Skewness        -0.086737
Kurtosis        -0.850644
```

Extracción de un vector dependiendo del valor de dos factores.

Imaginemos que tenemos un data.frame, con varios niveles. Y nos interesa hacer un doble filtrado, es decir extraer los valores de una variable dependiendo del resultado de otras dos.

Vamos a utilizar un data.frame hipotético llamado "datosmr", el cual tiene la siguiente estructura;

```
> str(datosmr)
'data.frame':	180 obs. of  4 variables:
 $ zona  : Factor w/ 5 levels "A","B","C","D",..: 1 1 1 1 1 1 2 2 2 2 ...
 $ forma : Factor w/ 2 levels "cuad","rect": 1 1 1 1 1 1 1 1 1 1 ...
 $ edad  : Factor w/ 3 levels "g","m","p": 3 3 3 3 3 3 3 3 3 3 ...
 $ conteo: num  0 7 3 3 7 3 10 1 7 6 ...
```

Es un data.frame con 180 observaciones de 4 variables: zona, forma, edad y conteo.

Donde conteo es la única variable numérica mientras que el resto son nominales con los valores que se presentan en el esquema anterior (para forma los valores posibles son "cuad" y "rect")

Haciendo una exploración general con el comando summary(), obtenemos:

```
> summary(datosmr)
```

```
zona    forma    edad      conteo
 A:36   cuad:90   g:60   Min.   : 0.000
 B:36   rect:90   m:60   1st Qu.: 0.000
 C:30             p:60   Median : 1.000
 D:30                    Mean   : 4.183
 E:48                    3rd Qu.: 3.000
                         Max.   :47.000
```

Vemos que hay 90 valores de forma="cuad" . Tenemos también valores de conteo y edad.

Ahora bien, si deseamos saber cuales son las estadísticas descriptivas para "conteo" cuando el factor "edad"="p" y "forma"="cuad" , podemos hacer un doble filtrado.

Vaciando primero nuestros datos a un data.frame donde "forma"="cuad" y luego filtrando un segundo data.frame con "edad"="p".

1. Primero creamos un nuevo data.frame que se llame “cuad”, a través del comando:

```
> cuad=subset(datosmr,forma=="cuad")

          #verificamos el sumario de “cuad”

> summary(cuad)
```

```
zona     forma     edad       conteo
 A:18   cuad:90   g:30   Min.   : 0.000
 B:18   rect: 0   m:30   1st Qu.: 0.000
 C:15             p:30   Median : 0.000
 D:15                    Mean   : 3.556
 E:24                    3rd Qu.: 3.000
                         Max.   :36.000v
```

Vemos que ahora nuestro data.frame “cuad” sólo contiene los 90 valores donde “forma=cuad”. Podemos sacar un resumen de frecuencias para cada valor de edad con el comando: table()

```
> table(cuad$conteo,cuad$edad)
```

```
     g  m  p
 0  25 20  2
 1   3  6  2
 2   1  2  1
 3   1  1  5
 4   0  0  1
 6   0  0  1
 7   0  0  3
 10  0  0  6
 12  0  0  1
 13  0  1  0
 14  0  0  1
 15  0  0  2
 16  0  0  1
 17  0  0  1
 20  0  0  1
 31  0  0  1
 36  0  0  1
```

Sin embargo si nos interesan estadísticas más exhaustivas, debemos de hacer un filtrado más para cada valor de edad, de la siguiente manera:

```
> cuad.p=subset(cuad,edad=="p")
> summary(cuad.p)
```

```
zona   forma     edad       conteo
 A:6   cuad:30   g: 0   Min.   : 0.000
 B:6   rect: 0   m: 0   1st Qu.: 3.000
 C:5             p:30   Median : 8.500
 D:5                    Mean   : 9.533
 E:8                    3rd Qu.:13.500
                        Max.   :36.000
```

Esto nos permite correr comandos como stem() o hist() que son para un vector.

```
> stem(cuad.p$conteo)
```

```
 The decimal point is 1 digit(s) to
the right of the |

  0 | 00112333334
  0 | 6777
  1 | 00000024
  1 | 5567
  2 | 0
  2 |
  3 | 1
  3 | 6
```

```
> hist(cuad.p$conteo)
```

El comando completo para salvar la gráfica es:

```
> jpeg(filename="hist.jpg")
> hist(cuad.p$conteo)
> dev.off()
```





> Estadística descriptiva: Se dedica a la descripción, visualización y resumen de datos originados a partir de los fenómenos de estudio.
>
> — WikiP

9. Descripción de grupos. Estadística descriptiva.

Descripción de grupos, ¿Por donde empezar?

De igual manera que antes de iniciar un análisis de datos, debemos preparar la base de datos para poder ser analizada (“4.Preparando la base de datos”, pág.22), antes de poder correr pruebas de diferencia o correlación, debemos describir de manera general al grupo.

Antes de poder correr pruebas de diferencia o correlación, debemos describir de manera general al grupo.

Esto abarca la elaboración de cuadros y gráficas, distribuciones de frecuencias, etc.

Lo primero que debemos hacer:

- Iniciar con un recuento de datos.
- Elaboración de distribuciones de frecuencias.
- Diagramas de tallo y hojas.
- Gráficas como el diagrama de barras o histogramas.

Si solo nombramos nuestra base de datos o data.frame, nos listará todos y cada uno de los datos. Esto no es problema si son 50 datos como en este caso, pero podría ser realmente complicado si la base es muy voluminosa, unos 25,000 casos o más, para esto es conveniente emplear los comandos head() y tail().

head() y tail()

Por esto es conveniente iniciar por los comandos head() y tail(). El comando head() nos muestra las primeras 6 filas del inicio y el comando tail() las 6 últimas filas.

> head(cars)	<pre> speed dist 1 4 2 2 4 10 3 7 4 4 7 22 5 8 16 6 9 10</pre>
> tail(cars)	<pre> speed dist 45 23 54 46 24 70 47 24 92 48 24 93 49 24 120 50 25 85</pre>

Esto nos indica que estamos trabajando con variables numéricas, “speed” y “dist”
Al analizar de igual manera la base “mtcars” (incluida en el R base), encontramos:

```
> head(mtcars)
                   mpg cyl disp  hp drat    wt  qsec vs am gear carb
Mazda RX4         21.0   6  160 110 3.90 2.620 16.46  0  1    4    4
Mazda RX4 Wag     21.0   6  160 110 3.90 2.875 17.02  0  1    4    4
Datsun 710        22.8   4  108  93 3.85 2.320 18.61  1  1    4    1
Hornet 4 Drive    21.4   6  258 110 3.08 3.215 19.44  1  0    3    1
Hornet Sportabout 18.7   8  360 175 3.15 3.440 17.02  0  0    3    2
Valiant           18.1   6  225 105 2.76 3.460 20.22  1  0    3    1
```

```
> tail(mtcars)
                mpg cyl  disp  hp drat    wt qsec vs am gear carb
Porsche 914-2  26.0   4 120.3  91 4.43 2.140 16.7  0  1    5    2
Lotus Europa   30.4   4  95.1 113 3.77 1.513 16.9  1  1    5    2
Ford Pantera L 15.8   8 351.0 264 4.22 3.170 14.5  0  1    5    4
Ferrari Dino   19.7   6 145.0 175 3.62 2.770 15.5  0  1    5    6
Maserati Bora  15.0   8 301.0 335 3.54 3.570 14.6  0  1    5    8
Volvo 142E     21.4   4 121.0 109 4.11 2.780 18.6  1  1    4    2
```

table()

Un recuento general de los datos lo podemos hacer con el comando table().

Esto produce un listado con los valores en la primera fila y sus frecuencias en la segunda.

En el caso de que estemos trabajando una variable categórica, el recuento es muy útil.

Como por ejemplo si usamos la base "mtcars" y resumimos los resultados por número de cilindros, así:

> table(mtcars$cyl)	4 6 8 11 7 14

Esto nos indica que en la categoría de 4 cilindros tenemos una frecuencia de 11, en la de 6 cilindros una frecuencia de 7 y en la categoría de 8 una frecuencia de 14

Lo que ya nos permite elaborar un cuadro de datos.

```
----------------------
Cilindros      frec
----------------------
     4         11
     6          7
     8         14
----------------------
               32
----------------------
```

Podemos de igual manera calcular los porcentajes por variable a través de:

table(base$variable) / length(base$variable)

> table(mtcars$cyl)/length(mtcars$cyl)	4 6 8 0.34375 0.21875 0.43750

Esto nos permite complementar nuestro cuadro:

```
---------------------------------------------
Cilindros      frec        fr        %
---------------------------------------------
     4          11        0.34375   34.375
     6           7        0.21875   21.875
     8          14        0.43750   43.750
---------------------------------------------
                32        1         100
---------------------------------------------
```

Sin embargo, si utilizamos el comando table() con la base “cars” y la variable “speed” obtenemos lo siguiente:

```
> table(cars$speed)

 4  7  8  9 10 11 12 13 14 15 16 17 18 19 20 22 23 24 25
 2  2  1  1  3  2  4  4  4  3  2  3  4  3  5  1  1  4  1
```

Como podemos observar, el recuento no nos ayuda mucho en este caso, por ser una variable cuantitativa.

Para estos casos es más práctico utilizar el comando stem() que nos produce un diagrama de tallo y hojas.

stem()

El comando stem(), produce un diagrama de tallo y hojas, que resulta particularmente útil para la elaboración de distribuciones de frecuencias.

```
stem(base$variable)
```

> stem(cars$speed)	The decimal point is at the | 4 | 00 6 | 00 8 | 00 10 | 00000 12 | 00000000 14 | 0000000 16 | 00000 18 | 0000000 20 | 00000 22 | 00 24 | 00000

Esto nos permite elaborar una distribución de frecuencias,

```
---------------------
Intervalos   frec
---------------------
 4 -  5.9      2
 6 -  7.9      2
 8 -  9.9      2
10 - 11.9      5
12 - 13.9      8
14 - 15.9      7
16 - 17.9      5
18 - 19.9      7
20 - 21.9      5
22 - 23.9      2
24 - 25.9      5
---------------------
              50
---------------------
```

basicStats

Una de las maneras más sencillas de producir las estadísticas descriptivas de un grupo es con el comando "basicStats" del paquete "fBasics".

Para poder utilizar este comando, es necesario instalar el paquete "fBasics"

```
>install.packages(fBasics)
```

Puede revisarse la sección "Instalación de Librerías", pág.19.

Una vez instalado el paquete, cargamos la librería:

>library(fBasics) Loading required package: timeDate Loading required package: timeSeries	Rmetrics Package fBasics Analysing Markets and calculating Basic Statistics Copyright (C) 2005-2014 Rmetrics Association Zurich Educational Software for Financial Engineering and Computational Science Rmetrics is free software and comes with ABSOLUTELY NO WARRANTY. https://www.rmetrics.org --- Mail to: info@rmetrics.org >

Ya cargada la librería, podemos utilizar el comando "basicStats" para obtener un resumen completo de las estadísticas descriptivas.

El comando "basicStats" corre el análisis en un vector. Por lo que si tenemos un data.frame, debemos extraer la variable a analizar a un vector para poder correrlo. **(Ver . "Extracción de una variable conociendo el nombre de la misma. (Extracción de un vector)", página 47)**

Ejemplo:

Vamos a utilizar la data.frame "cars" incluida en r base, para que puedan correrse los comandos presentados en los ejemplos.

Este data.frame incluye las variables "speed" y "dist" .

Como ya se mencionó anteriormente podemos hacer una exploración general del data.frame "cars", utilizando los comandos View(), head() y tail().

> head(cars)	<pre> speed dist 1 4 2 2 4 10 3 7 4 4 7 22 5 8 16 6 9 10</pre>
> tail(cars)	<pre> speed dist 45 23 54 46 24 70 47 24 92 48 24 93 49 24 120 50 25 85</pre>

Aplicamos el comando:

basicStats(base$variable)

> basicStats(cars$speed)	<pre> X..cars.speed nobs 50.000000 NAs 0.000000 Minimum 4.000000 Maximum 25.000000 1. Quartile 12.000000 3. Quartile 19.000000 Mean 15.400000 Median 15.000000 Sum 770.000000 SE Mean 0.747786 LCL Mean 13.897268 UCL Mean 16.902732 Variance 27.959184 Stdev 5.287644 Skewness -0.110553 Kurtosis -0.673092</pre>

En el siguiente cuadro se presentan las salidas del comando basicStats() y la traducción al español de cada una de las líneas que presenta.

nobs 50.000000	nobs n de observaciones
NAs 0.000000	NAs n de datos que no aplica
Minimum 4.000000	Minimum mínimo
Maximum 25.000000	Maximum máximo
1. Quartile 12.000000	1. Quartile primer cuartil
3. Quartile 19.000000	3. Quartile tercer cuartil
Mean 15.400000	Mean media aritmética
Median 15.000000	Median mediana
Sum 770.000000	Sum sumatoria de datos
SE Mean 0.747786	SE Mean error estándar de la media
LCL Mean 13.897268	LCL Mean Interva. de Confi. inf.
UCL Mean 16.902732	UCL Mean Interva. de Confi. sup.
Variance 27.959184	Variance varianza
Stdev 5.287644	Stdev desviación estándar
Skewness -0.110553	Skewness sesgo
Kurtosis -0.673092	Kurtosis curtósis

Es conveniente recordar que los límites para el sesgo y la curtosis son dos veces el error estándar del sesgo(ses) y dos veces el error estándar de la curtosis (sek) respectivamente. Y que un grupo que es simétrico y mesocúrtico se considera aproximadamente normal.

Las fórmulas para el ses y el sek son respectivamente: ses=√6/n y sek=√24/n

Para el presente ejemplo:

Los límites para el sesgo serían: 2(√6/50) = 2*(6/50)^0.5 = 0.6928203

Y los límites para la curtosis serían: 2(√24/50) = 2*(24/50)^0.5 = 0.6928203 = 1.385641

Por lo que el grupo puede considerarse normal.

Los límites para la curtosis también pueden calcularse multiplicando el ses por 4 es decir:

4(√6/50) = 4*(6/50)^0.5 = 1.385641

Cálculo individual de estadísticas descriptivas

Si se desean calcular las estadísticas descriptivas de manera individual, ya sea para cálculos individuales o para un ejercicio específico, podemos obtenerlas a través de los siguientes comandos.

Siendo "d" los siguientes valores: `d=c(12,8,15,17,13,20,20)`

1. Media aritmética , mean()

```
>mean(d)
[1] 15
```

2. Mediana , median()

```
>median(d)
[1] 15
```

3. Moda.

No existe ninguna función que nos brinde la moda de manera directa, podemos encontrar una función "Mode" para su cálculo en el paquete "prettyR" , pero no es imprescindible. Tomando en cuenta que la moda es el punto en la escala de medición donde se ubica la mayor frecuencia, solo debemos determinar la frecuencia de los datos para poder estimar la moda.

Una forma sencilla de hacer esto es con listar las frecuencias a través del comando table()

```
> table(d)
d
 8 12 13 15 17 20
 1  1  1  1  1  2
```

Con esto podemos observar que la máxima frecuencia se encuentra en el valor 20, por lo que la moda es 20.

4. Varianza , var()

```
> var(d)
[1] 19.33333
```

5. Desviación estándar, sd()

```
> sd(d)
[1] 4.396969
```

6. Sesgo , Skewness() y curtosis , kurtosis() - a través de la librería "fBasics"

Para poder usar el sesgo y la curtosis sin tener que hacer los cálculos, podemos instalar el paquete "fBasics" y cargar la librería "fBasics".

Puede revisarse la sección "Instalación de Librerías", pág.19.

>library(fBasics) Loading required package: timeDate Loading required package: timeSeries	Rmetrics Package fBasics Analysing Markets and calculating Basic Statistics Copyright (C) 2005-2014 Rmetrics Association Zurich Educational Software for Financial Engineering and Computational Science Rmetrics is free software and comes with ABSOLUTELY NO WARRANTY. https://www.rmetrics.org --- Mail to: info@rmetrics.org >

Una vez cargada la librería, ya podemos correr los comandos para el sesgo y la curtosis

```
> skewness(d)
[1] -0.2016615
attr(,"method")
[1] "moment"

> kurtosis(d)
[1] -1.561406
attr(,"method")
[1] "excess"
```

Los límites para el sesgo y la curtosis se pueden estimar con :

Límites para el sesgo es 2 veces el ses. El ses=√(6/n)

Límites para la curtosis es 2 veces el sek. El sek=√(24/n).
También podemos estimar los límites para el sesgo, al multiplicar el ses por 4, es decir 4 veces el ses.

El número de elementos en el vector lo podemos determinar a través de la función length().
De esta forma calculamos el ses y estimamos los límites, así:

```
> ses=sqrt(6/length(d))
> ses*2                  #los límites del sesgo son 2 veces el ses
[1] 1.85164              #límites del sesgo

> sek=sqrt(24/length(d))
> sek*2                  #los límites de la curtosis son 2 veces el sek
[1] 3.70328              #límites de la curtosis
```

```
> ses*4                    #otra forma de estimar los límites para la curtosis
[1] 3.70328                #límites de la curtosis
```

7. Cinco números de resumen, summary()

```
> summary(d)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
    8.0    12.5    15.0    15.0    18.5    20.0 
```

Con esto podemos calcular el mínimo, el máximo, el rango, el rango intercuartil y la desviación cuartil.

El rango sería la diferencia entre el máximo y el mínimo. Podemos obtenerlo también desde el comando range()

```
> range(d)
[1] 8 20
```

El rango intercuartil es la diferencia entre el tercer cuartil y el primer cuartil.

La desviación cuartil es el rango intercuartil dividido dos.

Debemos recordar, que cuando el grupo es asimétrico, la tendencia central que se reporta es la mediana y la medida de dispersión es la desviación cuartil.

8. Prueba de normalidad de los datos, shapiro.test()
Fabricamos un grupo llamado "datos", constituido por 60 números aleatorios con una media aritmética de 50 y una desviación estándar de 10.

```
> datos=c(rnorm(60,50,10))
> shapiro.test(datos)

     Shapiro-Wilk normality test

data:  datos
W = 0.98362, p-value = 0.5988
```

Un valor de p < 0.05, indica que existe diferencia significativa de la normalidad.

En este caso el valor de p=0.5988 (> 0.05) nos indica que no hay diferencia significativa de la normalidad.

9. Prueba de varianzas iguales, var.test()

Comparamos dos grupos aleatorios con distribución normal, con media de 50 y desviación estándar de 10, llamados “datos” y “datos2”

```
> var.test(datos,datos2)

     F test to compare two variances

data:  datos and datos2
F = 1.0241, num df = 59, denom df = 59, p-value = 0.9275
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.6116975 1.7144154
sample estimates:
ratio of variances
          1.024062
```

10. Otros comandos útiles para la descripción de los grupos.

- stem() produce un diagrama de tallo y hojas de los datos.
- quantile() produce el cálculo de cuartiles.
- hist() produce un histograma de los datos.
- boxplot() produce un diagrama de caja y bigotes de los datos.
- ls() muestra los objetos en memoria



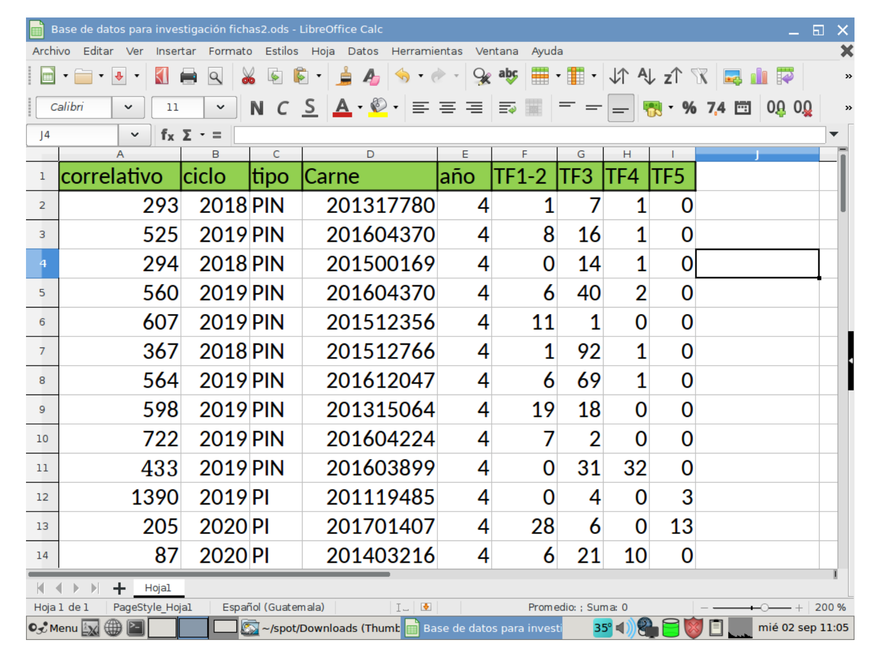
10. Un ejemplo de estadística descriptiva de principio a fin.

A menos que utilicemos R para hacer análisis con cierta frecuencia, es normal que se nos olviden ciertos comandos o tengamos problemas al momento de empezar a analizar un grupo de datos.

Esto es especialmente verdadero cuando empleamos R en nuestros estudios en la Universidad y en las investigaciones que se desarrollan en los distintos cursos, las cuales están rodeadas de muchos más conocimientos y actividades que hacen que dificultan el tener presente todos los comandos y pautas importantes que se han mencionado.

Para ayudar a ese nuevo “arranque en frío” del proceso de análisis, voy a presentar un ejemplo paso a paso del análisis de la estadística descriptiva en un grupo y la elaboración de un reporte preliminar, de manera que sirva de orientación y en la medida de lo posible no haga necesaria la lectura de todos los capítulos del libro.

Es necesario remarcar que en investigación no se pueden emplear “recetas” para el análisis y que el criterio del investigador es quizás la herramienta más importante a tomar en cuenta, por lo que este ejemplo no busca reducir el proceso de investigación a la aplicación de una receta, sino más bien orientar los primeros pasos en la descripción de un grupo, manteniendo presente que la interpretación de los datos debe ser realizada teniendo presente el contexto de la investigación, los objetivos y las limitaciones encontradas durante el trabajo de campo y análisis de los datos.

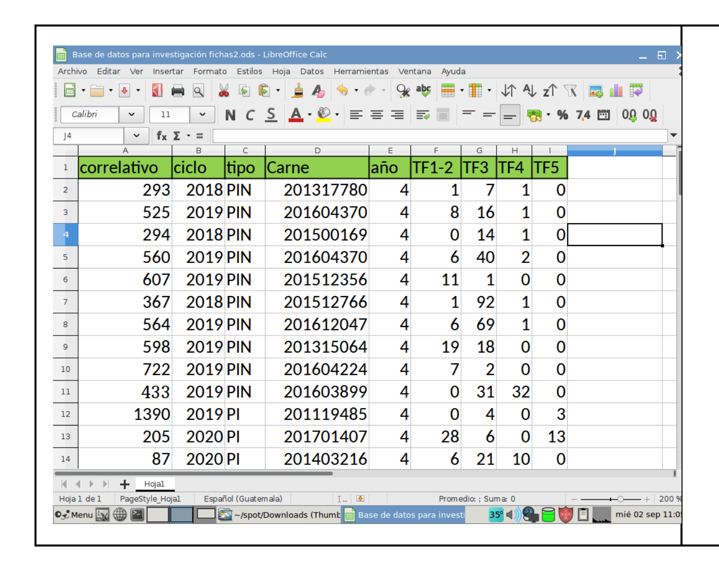


correlativo	ciclo	tipo	Carne	año	TF1-2	TF3	TF4	TF5
293	2018	PIN	201317780	4	1	7	1	0
525	2019	PIN	201604370	4	8	16	1	0
294	2018	PIN	201500169	4	0	14	1	0
560	2019	PIN	201604370	4	6	40	2	0
607	2019	PIN	201512356	4	11	1	0	0
367	2018	PIN	201512766	4	1	92	1	0
564	2019	PIN	201612047	4	6	69	1	0
598	2019	PIN	201315064	4	19	18	0	0
722	2019	PIN	201604224	4	7	2	0	0
433	2019	PIN	201603899	4	0	31	32	0
1390	2019	PI	201119485	4	0	4	0	3
205	2020	PI	201701407	4	28	6	0	13
87	2020	PI	201403216	4	6	21	10	0

Partamos de un caso concreto:

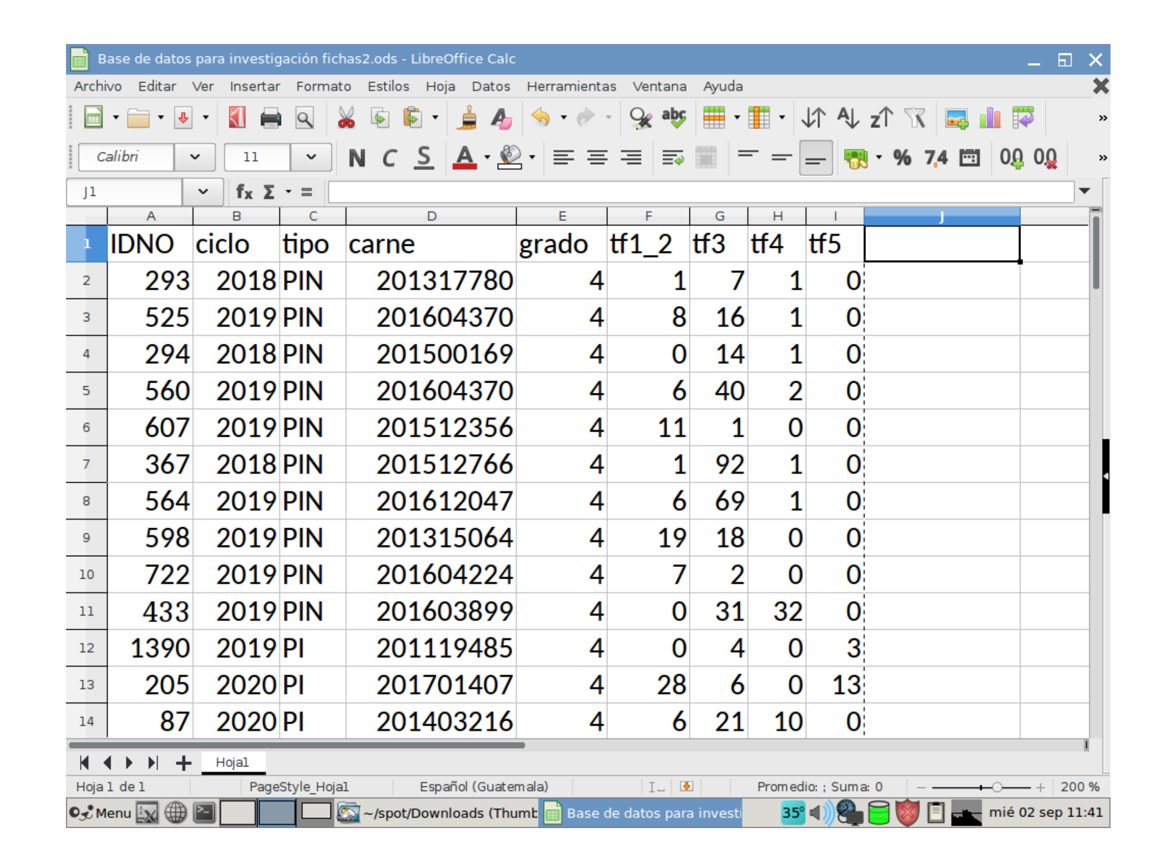
Se realiza una investigación sobre el tiempo que le toma a unos estudiantes de odontología finalizar ciertas fases del proceso de ingreso de pacientes a una clínica en particular.
El objetivo es hacer una descripción inicial de los datos recolectados en tres años distintos, el 2018, 2019 y 2020. El tiempo se registra en días y las Fases del Proceso se identifican como Fase 1 y 2, Fase 3, Fase 4 y Fase 5.
Otras variables a tomar en cuenta, son el tipo de paciente y el año que cursa el estudiante.

A. Arreglo de la base de datos



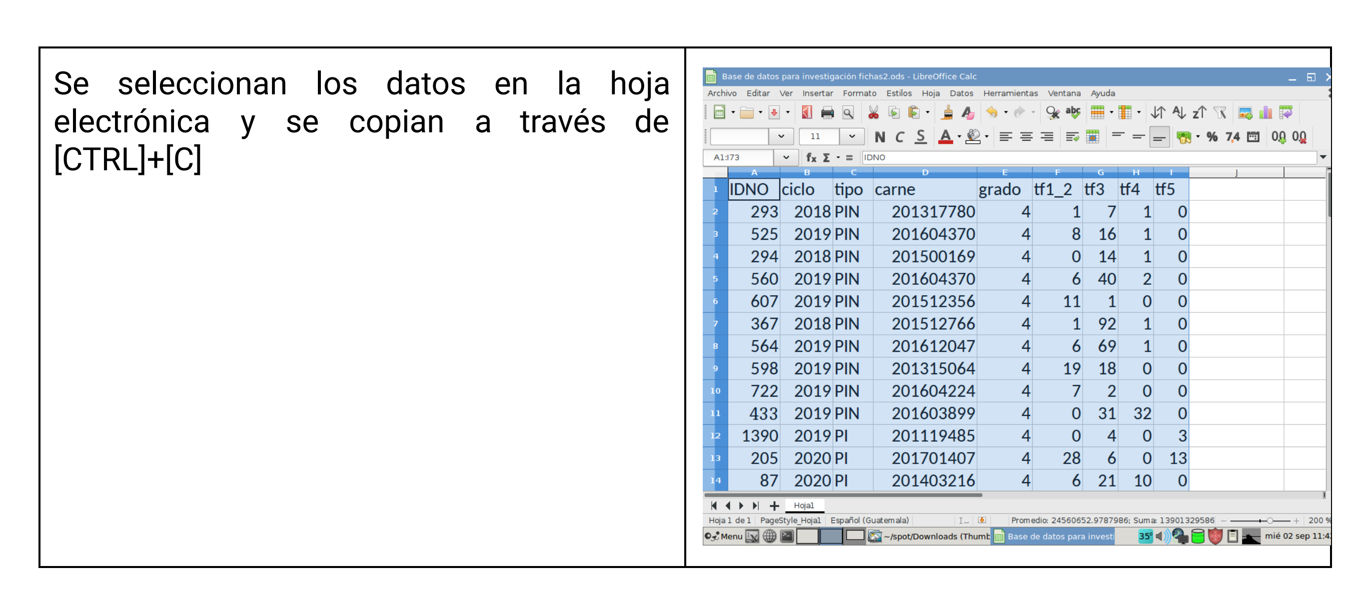
correlativo	ciclo	tipo	Carne	año	TF1-2	TF3	TF4	TF5
293	2018	PIN	201317780	4	1	7	1	0
525	2019	PIN	201604370	4	8	16	1	0
294	2018	PIN	201500169	4	0	14	1	0
560	2019	PIN	201604370	4	6	40	2	0
607	2019	PIN	201512356	4	11	1	0	0
367	2018	PIN	201512766	4	1	92	1	0
564	2019	PIN	201612047	4	6	69	1	0
598	2019	PIN	201315064	4	19	18	0	0
722	2019	PIN	201604224	4	7	2	0	0
433	2019	PIN	201603899	4	0	31	32	0
1390	2019	PI	201119485	4	0	4	0	3
205	2020	PI	201701407	4	28	6	0	13
87	2020	PI	201403216	4	6	21	10	0

- Eliminar formatos innecesarios.
- Renombrar variables según las normas de R.
- Marcar espacios vacíos con “NA”



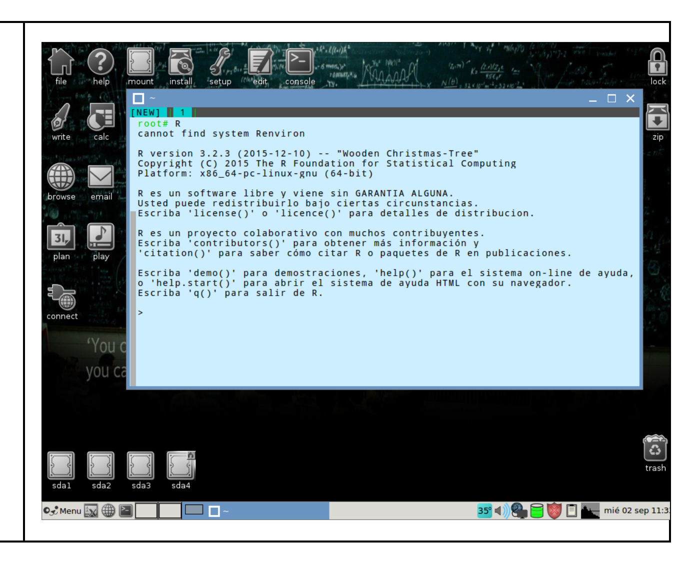
IDNO	ciclo	tipo	carne	grado	tf1_2	tf3	tf4	tf5
293	2018	PIN	201317780	4	1	7	1	0
525	2019	PIN	201604370	4	8	16	1	0
294	2018	PIN	201500169	4	0	14	1	0
560	2019	PIN	201604370	4	6	40	2	0
607	2019	PIN	201512356	4	11	1	0	0
367	2018	PIN	201512766	4	1	92	1	0
564	2019	PIN	201612047	4	6	69	1	0
598	2019	PIN	201315064	4	19	18	0	0
722	2019	PIN	201604224	4	7	2	0	0
433	2019	PIN	201603899	4	0	31	32	0
1390	2019	PI	201119485	4	0	4	0	3
205	2020	PI	201701407	4	28	6	0	13
87	2020	PI	201403216	4	6	21	10	0

B. Ingreso de la base de datos a R.



Se seleccionan los datos en la hoja electrónica y se copian a través de [CTRL]+[C]

IDNO	ciclo	tipo	carne	grado	tf1_2	tf3	tf4	tf5
293	2018	PIN	201317780	4	1	7	1	0
525	2019	PIN	201604370	4	8	16	1	0
294	2018	PIN	201500169	4	0	14	1	0
560	2019	PIN	201604370	4	6	40	2	0
607	2019	PIN	201512356	4	11	1	0	0
367	2018	PIN	201512766	4	1	92	1	0
564	2019	PIN	201612047	4	6	69	1	0
598	2019	PIN	201315064	4	19	18	0	0
722	2019	PIN	201604224	4	7	2	0	0
433	2019	PIN	201603899	4	0	31	32	0
1390	2019	PI	201119485	4	0	4	0	3
205	2020	PI	201701407	4	28	6	0	13
87	2020	PI	201403216	4	6	21	10	0

<table>
<tr>
<td>Se inicia R desde la ventana de comandos</td>
<td>
</td>
</tr>
</table>

Se ingresan los datos copiados de la hoja electrónica a un **data.frame** que vamos a llamar **fichas**, utilizando el comando **read.delim** la sintaxis completa es:

```
nombre=read.delim(“clipboard”)
```

Donde “nombre” será **fichas**.

<table>
<tr>
<td>>fichas=read.delim(“clipboard”)
></td>
<td>[NEW] 1
> fichas=read.delim("clipboard")
></td>
</tr>
</table>

Verificamos la base de datos con head()

head(fichas)	`> head(fichas)` `  IDNO ciclo tipo     carne grado tf1_2 tf3 tf4 tf5` `1  293  2018  PIN 201317780     4     1   7   1   0` `2  525  2019  PIN 201604370     4     8  16   1   0` `3  294  2018  PIN 201500169     4     0  14   1   0` `4  560  2019  PIN 201604370     4     6  40   2   0` `5  607  2019  PIN 201512356     4    11   1   0   0` `6  367  2018  PIN 201512766     4     1  92   1   0` `>`

C. Revisión general de los datos summary(), table() y basicStats()

Una visión general de los datos la obtenemos a través del comando summary()

```
> summary(fichas)
      IDNO             ciclo        tipo        carnet             grado
 Min.   :  10.0   Min.   :2018   PI :46   Min.   :201119485   3  :
2
 1st Qu.: 256.0   1st Qu.:2019   PIN:22   1st Qu.:201315064   4
:44
 Median : 585.5   Median :2019   PT : 4   Median :201512356   5
:19
 Mean   : 682.4   Mean   :2019            Mean   :201465655   PRC:
7
 3rd Qu.:1074.2   3rd Qu.:2019            3rd Qu.:201604224
 Max.   :2104.0   Max.   :2020            Max.   :201701447
                                          NA's   :3
     tf1_2            tf3               tf4               tf5
 Min.   : 0.00   Min.   :  0.00   Min.   : 0.000   Min.   :  0.00
 1st Qu.: 0.00   1st Qu.:  2.75   1st Qu.: 0.000   1st Qu.:  0.00
 Median : 3.50   Median : 10.00   Median : 1.000   Median :  2.00
 Mean   : 6.25   Mean   : 19.71   Mean   : 5.486   Mean   : 32.28
 3rd Qu.: 8.00   3rd Qu.: 21.50   3rd Qu.: 7.000   3rd Qu.: 25.25
 Max.   :30.00   Max.   :238.00   Max.   :58.000   Max.   :273.00
```

Podemos observar que las estadísticas descriptivas de las variables “IDNO”, “ciclo” y “carnet” no tienen significado. En el caso de “IDNO” solo es el número de identificación y por lo tanto es una variable nominal por lo que carece de características numéricas y simplemente ignoramos estos datos. Lo mismo sucede con la variable “carnet”, al ser el número de carné del estudiante es una variable nominal que no tiene características numéricas.

La variable “ciclo” también es una variable nominal, por lo que no son valores numéricos, pero a diferencia de “IDNO” y “carnet”, en este caso si nos interesa saber cuántas unidades de estudio se ubican en cada ciclo, por lo que haremos uso del comando **table()** para obtener esta información.

> table(fichas$ciclo)	`2018 2019 2020` `  13   41   18` `>`

Esto ya nos permite extraer información preliminar de los datos:

`> table(fichas$ciclo)` `2018 2019 2020` `  13   41   18`	Los datos están mayormente constituidos por fichas de pacientes del año 2019 con 41
`>` `prop.table(table(fichas$ciclo))` `     2018      2019      2020` `0.1805556 0.5694444 0.2500000`	Vemos que el % por ciclo es: 2018 18.06% 2019 56.94% 2020 25.00%
`tipo` `PI :46` `PIN:22` `PT : 4`	Que el tipo de paciente tiene una frecuencia de: Paciente integral 46 Paciente integral niño 22 Paciente de Total 4 Por lo que los resultados no reflejan el comportamiento de los pacientes de Total y debería aumentarse el número de pacientes en este renglón o retirarlos del estudio.

grado 3 : 2 4 :44 5 :19 PRC: 7	El grupo estuvo constituido mayormente por alumnos de 4o y 5o grado. Habiendo únicamente 2 pacientes de 3o y 7 de Pendientes de Requisitos Clínicos, por lo que debería aumentarse la recolección en estos rubros o retirarlos del estudio.
tf1_2 Min. : 0.00 1st Qu.: 0.00 Median : 3.50 Mean : 6.25 3rd Qu.: 8.00 Max. :30.00	Con respecto al tiempo necesario para finalizar la fase 1 y 2, encontramos una media de 6.25 y mediana de 3.5 Datos que son aplicables, pero insuficientes para un adecuado análisis.

Para poder hacer un mejor análisis de las variables “tf1_2”, “tf3”, “tf4” y “tf5” vamos a cargar la librería “fBasics” y usar el comando “basicStats”

Cargamos la librería “fBasics”

> library(fBasics)	Loading required package: timeDate Loading required package: timeSeries Rmetrics Package fBasics Analysing Markets and calculating Basic Statistics Copyright (C) 2005-2014 Rmetrics Association Zurich Educational Software for Financial Engineering and Computational Science Rmetrics is free software and comes with ABSOLUTELY NO WARRANTY. https://www.rmetrics.org --- Mail to: info@rmetrics.org >

y usamos el comando "basicStats" con cada variable

> basicStats(fichas$tf1_2)	X..fichas.tf1_2 nobs 72.000000 NAs 0.000000 Minimum 0.000000 Maximum 30.000000 1. Quartile 0.000000 3. Quartile 8.000000 Mean 6.250000 Median 3.500000 Sum 450.000000 SE Mean 0.921137 LCL Mean 4.413305 UCL Mean 8.086695 Variance 61.091549 Stdev 7.816108 Skewness 1.468003 Kurtosis 1.391960

> basicStats(fichas$tf3)	X..fichas.tf3 nobs 72.000000 NAs 0.000000 Minimum 0.000000 Maximum 238.000000 1. Quartile 2.750000 3. Quartile 21.500000 Mean 19.708333 Median 10.000000 Sum 1419.000000 SE Mean 4.032557 LCL Mean 11.667643 UCL Mean 27.749024 Variance 1170.829225 Stdev 34.217382 Skewness 4.236162 Kurtosis 21.846132

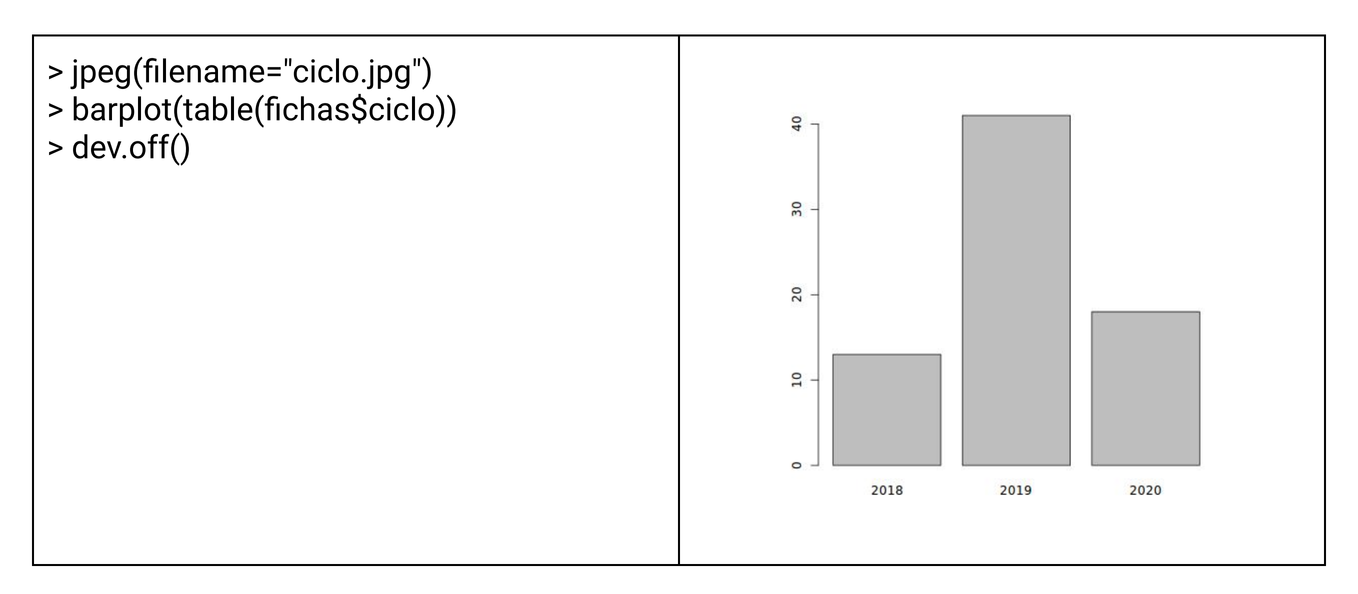
```
> basicStats(fichas$tf4)
```

```
              X..fichas.tf4
nobs              72.000000
NAs                0.000000
Minimum            0.000000
Maximum           58.000000
1. Quartile        0.000000
3. Quartile        7.000000
Mean               5.486111
Median             1.000000
Sum              395.000000
SE Mean            1.262650
LCL Mean           2.968458
UCL Mean           8.003764
Variance         114.788537
Stdev             10.713941
Skewness           2.772490
Kurtosis           8.224687
```

D. Gráficas exploratorias.

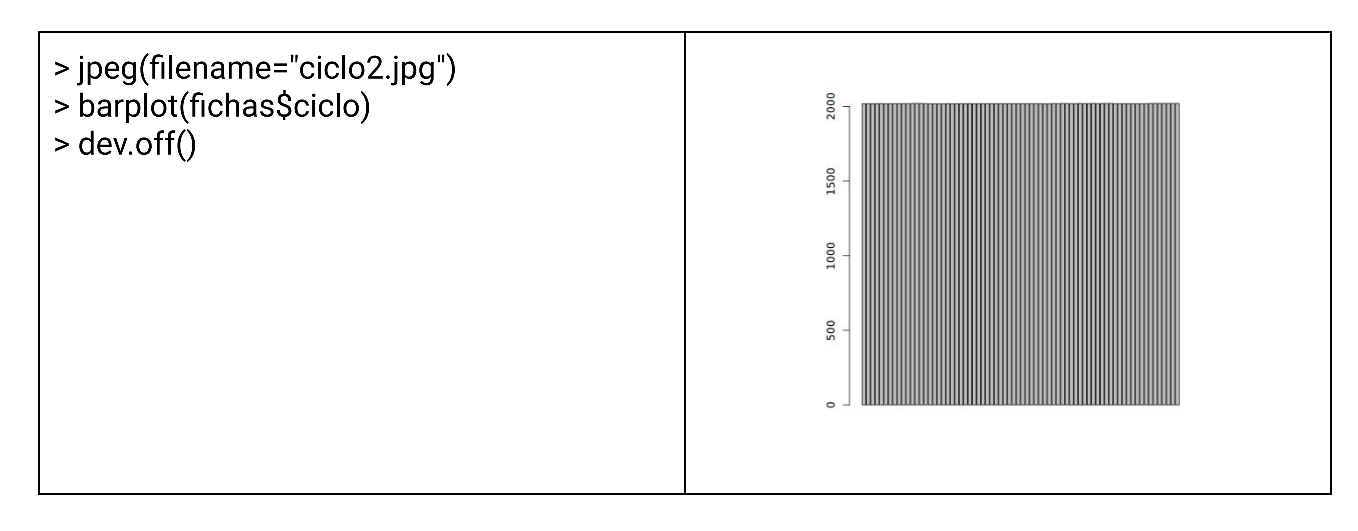
Podemos observar también el comportamiento general de los datos a través de diagrama de barras o histogramas, para esto:

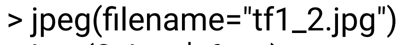
- Generamos gráficas de cada variable.
- Guardamos aquellas gráficas que resulten convenientes, según lo visto en la sección “6. Archivo de salidas, Guardando Gráficas”, “Archivo gráfico jpg . jpg()” página 32.



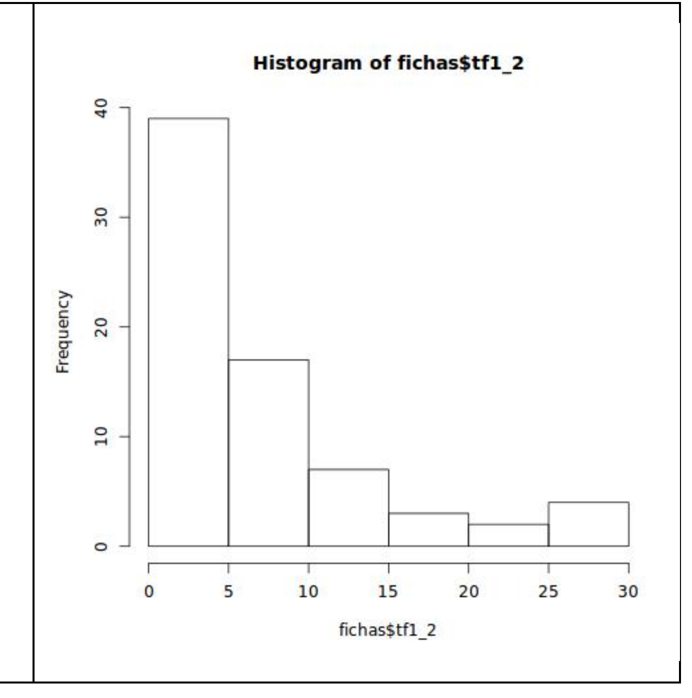
En este caso particular el comando barplot() que genera una gráfica de barras no lo aplicamos directamente a la variable, sino al resultado de table() para que resulte una gráfica que nos muestre la frecuencia por ciclo.

De aplicarlo de manera convencional, el programa considera que es una variable numérica y nos regresaría una gráfica con cada uno de los valores en la serie.

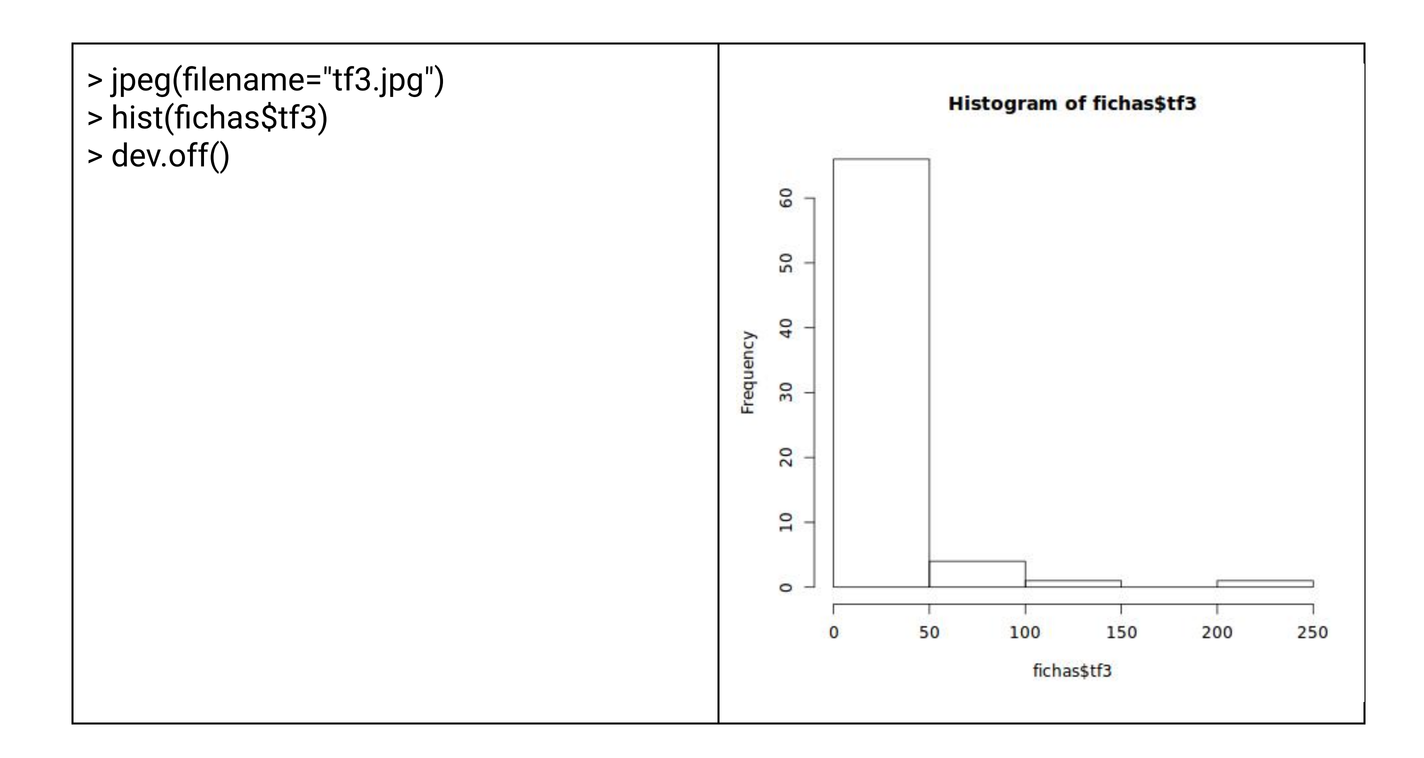




```
> jpeg(filename="tf1_2.jpg")
> hist(fichas$tf1_2)
> dev.off()
```



```
> jpeg(filename="tf3.jpg")
> hist(fichas$tf3)
> dev.off()
```



E. Variantes del valor de una variable dependiendo del grupo o factor.

Si queremos determinar si existe alguna variante del valor de una variable dependiendo del grupo, podemos aplicar el comando:

tapply()

Para determinar el valor de la media aritmética, dependiendo del grupo, utilizamos:

>tapply(datos$valor,datos$grupo,mean)

Para determinar el valor de la desviación estándar, dependiendo del grupo, utilizamos:

> tapply(datos$valor,datos$grupo,sd)

Si queremos observar el valor promedio de “tf1_2” dependiendo del ciclo, usamos: > tapply(fichas$tf1_2,fichas$ciclo,mean)	`2018 2019 2020` `5.461538 7.560976 3.833333`
Para observar el valor promedio de “tf1_2” dependiendo del grado, usamos: > tapply(fichas$tf1_2,fichas$grado,mean)	`3 4 5 PRC` `18.5000000 6.6818182 6.1052632 0.4285714`

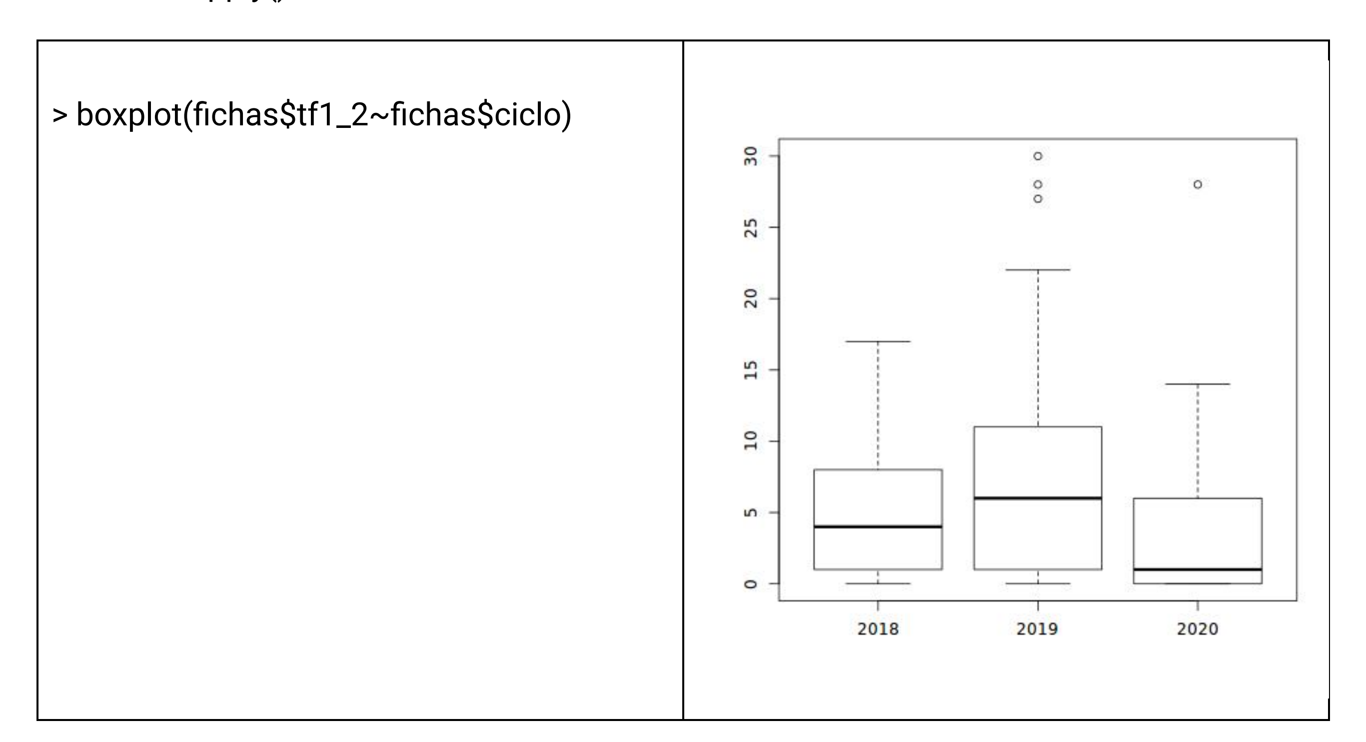
La estructura del comando se mantiene. Únicamente sustituimos el análisis a correr

tapply(datos$valor,datos$grupo,**mean**)

Podemos ver una serie de comandos para utilizar en **“14. Referencia de comandos útiles. Comandos para el análisis numérico de datos”**, página 101.

F. Gráficas exploratorias para los valores de la variable, dependiendo de un vector.

En algunos casos puede ser deseable el comparar de una manera gráfica algunos resultados dependiendo de un factor. De una forma similar a como empleamos el comando tapply() en la sección anterior.



G. Unas últimas palabras.

R es un lenguaje muy versátil y no existe un único camino para obtener los resultados deseados.

La presente guía es solamente una sugerencia y una orientación para tener presente en el caso de que necesitemos el apoyo al momento de iniciar con los análisis.

De hecho cada persona debe ir desarrollando su forma de trabajar, dependiendo de su facilidad de aprendizaje, su dominio del lenguaje y los objetivos e intereses propios.

Un ejemplo de la versatilidad de R y otra forma de generar gráficas similares a la anterior por otra ruta, es extrayendo los valores de tf1_2 por ciclo, y crear vectores independientes para cada ciclo, con el comando **subset()**.

De tal manera que creamos un vector con los resultados de “tf1_2” (la variable) cuando el valor del ciclo sea igual a 2018 (grupo). Creamos otro vector con los resultados de “tf1_2” cuando el valor del ciclo sea igual a 2019 y así sucesivamente.

```
> tf1_2.ciclo2018=subset(fichas$tf1_2,fichas$ciclo=="2018")
> tf1_2.ciclo2019=subset(fichas$tf1_2,fichas$ciclo=="2019")
> tf1_2.ciclo2020=subset(fichas$tf1_2,fichas$ciclo=="2020")
```

El nombre de los vectores puede ser cualquiera. Desde algo tan simple como “a” o “grupo1”, pero es recomendable que nos ayude a recordar qué fue lo que colocamos en cada vector creado.
Una vez creados los vectores, simplemente utilizamos el comando boxplot()

boxplot(vector_a, vector_b, vector_c)

En nuestro caso:

```
> jpeg(filename="tf1_2_ciclo.jpg")
> boxplot(tf1_2.ciclo2018,tf1_2.ciclo2019,tf1_2.ciclo2020)
> dev.off()
```

11. Pruebas de diferencia.



Las pruebas estadísticas detectan diferencias significativas entre grupos.

— WikiP

11. Pruebas de diferencia

t de Student para diferencia de dos muestras.

La t de Student es una prueba de contraste paramétrica para comprobar la igualdad de las medias de dos muestras o de una muestra contra el parámetro.

Para poder aplicarse la prueba t de Student debe cumplir con los siguientes supuestos:

- Nivel de medida de las variables: métricas, es decir, intervalo o razón.
- Distribución: normal o aproximadamente normal.
- Tipo de diseño: equilibrado o no equilibrado.
- Varianzas poblacionales: desconocidas, supuestamente iguales o sin supuesto de igualdad.
- Observaciones: aleatorias e independientes.
- Hipótesis que se somete a prueba: la diferencia entre las dos medias toma un determinado valor, generalmente cero.

Debe cumplirse con las escalas numéricas, el tipo de diseño, las observaciones independientes y el planteamiento de las hipótesis, en el diseño estadístico.

Hay que confirmar la normalidad en la distribución de los grupos y la igualdad de varianzas, para que la prueba de t tenga validez.

Para normalidad se puede emplear la prueba de Shapiro, con shapiro.test()
Para la igualdad de varianzas, con el test de varianzas iguales, var.test()

Así, teniendo las dos poblaciones “datos” y “datos2”, procedemos:

> shapiro.test(datos)	Shapiro-Wilk normality test data: datos W = 0.98362, p-value = 0.5988
> shapiro.test(datos2)	Shapiro-Wilk normality test data: datos2 W = 0.96446, p-value = 0.0779

> var.test(datos,datos2)	F test to compare two variances data: datos and datos2 F = 1.0241, num df = 59, denom df = 59, p-value = 0.9275 alternative hypothesis: true ratio of variances is not equal to 1 95 percent confidence interval: 0.6116975 1.7144154 sample estimates: ratio of variances 1.024062

Los resultados de normalidad son significativos al 95% (p-value > 0.05)
La igualdad de varianzas resultado positivo (p-value >0.05)
Por lo que se puede aplicar la prueba t de Student para determinar diferencia entre los grupos.

> t.test(datos,datos2)	Welch Two Sample t-test data: datos and datos2 t = 1.7641, df = 117.98, p-value = 0.0803 alternative hypothesis: true difference in means is not equal to 0 95 percent confidence interval: -0.3908077 6.7695402 sample estimates: mean of x mean of y 51.40990 48.22054

Como el valor de p es mayor a 0.05, no se encuentra diferencia significativa.

Si comparamos los grupos “datos” con “datos3”, para determinar diferencia, debemos probar la normalidad de “datos3”, la de “datos” había sido probada en el ejemplo anterior.

```
> shapiro.test(datos3)
```

```
     Shapiro-Wilk normality test

data:  datos3
W = 0.98056, p-value = 0.4525
```

El grupo es normal.

Comparamos las varianzas

```
> var.test(datos,datos3)
```

```
F test to compare two variances

data:  datos and datos3
F = 1.273, num df = 59, denom df = 59, p-value = 0.3565
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.7603895 2.1311572
sample estimates:
ratio of variances
          1.272992
```

Al comparar las varianzas, encontramos que no hay diferencia.
Por lo que podemos correr la prueba t de Student

```
> t.test(datos,datos3)
```

```
Welch Two Sample t-test

data:  datos and datos3
t = 10.739, df = 116.32, p-value < 2.2e-16
alternative hypothesis: true difference in
means is not equal to 0
95 percent confidence interval:
 15.04990 21.85659
sample estimates:
mean of x mean of y
 51.40990  32.95666
```

En este caso, encontramos que el valor de p es mucho menor a 0.05, por lo que podemos concluir una diferencia significativa.

t de Student para diferencia de una muestra.

Si se desea comprobar si una muestra difiere de la media poblacional, se puede emplear el comando:

t.test(vector, mu=media)

En el caso que quisiéramos comparar la muestra "datos" con la media poblacional de 40, procedemos así:

`> t.test(datos,mu=40)`	One Sample t-test data: datos t = 8.8726, df = 59, p-value = 1.864e-12 alternative hypothesis: true mean is not equal to 40 95 percent confidence interval: 48.83669 53.98312 sample estimates: mean of x 51.4099

Como el resultado de p es mucho menor a 0.05, se encuentra diferencia significativa entre "datos" y la población teórica con mu=40

Prueba U de Mann-Whitney, Prueba de Wilcoxon o Prueba de Rangos con signo de Wilcoxon, wilcox.test()

Si no se cumplen los supuestos de la prueba t de Student , la opción para la determinación de diferencia entre dos grupos es la prueba no paramétrica U de Mann-Whitney también llamada prueba de Wilcoxon.

La escala de la variable debe ser por lo menos ordinal.

El comando para el análisis es:

```
wilcox.test(x,y, paired=FALSE, conf.level=0.95)
```

```
>wilcox.test(datos,datos2, paired=FALSE,conf.level=0.95)

        Wilcoxon rank sum test with continuity correction

data:  datos and datos2
W = 2216, p-value = 0.0292
alternative hypothesis: true location shift is not equal to 0
```

El argumento “paired=FALSE” es el que indica que los datos provienen de muestras independientes.
Si los datos provienen de la misma población sería “paired=TRUE”

En este caso la prueba indica diferencia significativa con una $p<0.05$

Prueba de ANOVA, aov()

Del mismo modo que la t de Student, la prueba ANOVA es una prueba paramétrica y como tal requiere una serie de supuestos para poder ser aplicada correctamente.

Denominada ANOVA o análisis de la varianza, en realidad nos va a servir no solo para estudiar las dispersiones o varianzas de los grupos, sino para estudiar sus medias y la posibilidad de crear subconjuntos de grupos con medias iguales.

Se puede decir que la prueba ANOVA es la generalización de la t de Student, ya que si realizamos una prueba ANOVA en la comparación de solo dos grupos, obtenemos los mismos resultados.

Al igual que la t de Student, se requiere que cada uno de los grupos a comparar tenga distribuciones normales, o lo que es más exacto, que lo sean sus residuales.

Los residuales son las diferencias entre cada valor y la media de su grupo.

Además debemos estudiar la dispersión o varianzas de los grupos, es decir estudiar su homogeneidad.

Cuando mayor sean los tamaños de los grupos, menos importante es asegurar estos dos supuestos, ya que el ANOVA suele ser una técnica bastante “robusta” comportándose bien respecto a transgresiones de la normalidad.

No obstante, si tenemos grupos de tamaño inferior a 30, es importante estudiar la normalidad de los residuos para ver la conveniencia o no de utilizar el análisis de la varianza.

Si no fuera posible utilizar directamente el ANOVA, podemos recurrir al uso de pruebas no paramétricas, como la de Kruskal-Wallis.

Para la prueba de ANOVA, necesitaremos un data.frame con dos vectores, la variable de agrupación y la variable de valores.

> grupo=c(rep("A",30),rep("B",30),rep("C",30))	#creamos un vector con 30 datos de cada uno de los valores A, B y C.
> valor=c(rnorm(90,75,5))	#creamos 90 valores aleatorios con distribución normal, con media de 75 y desviación estándar de 5.
> ej1=data.frame(grupo,valor)	#creamos el data.frame "ej1" combinando "grupo" y "valor"
> head(ej1) grupo valor 1 A 78.23042 2 A 81.76152 3 A 79.96561 4 A 68.30342 5 A 68.49030 6 A 75.97284	# verificamos la estructura del data.frame "ej1", mostrando las primeras seis filas.

Para verificar la normalidad en cada grupo, los extraemos a sendos vectores con el comando subset():

```
grupo.a=subset(ej1$valor,ej1$grupo==”A”)
grupo.b=subset(ej1$valor,ej1$grupo==”B”)
grupo.c=subset(ej1$valor,ej1$grupo==”C”)
```

> shapiro.test(grupo.a) Shapiro-Wilk normality test data: grupo.a W = 0.98397, p-value = 0.9184	> shapiro.test(grupo.b) Shapiro-Wilk normality test data: grupo.b W = 0.96436, p-value = 0.3983
> shapiro.test(grupo.c) Shapiro-Wilk normality test data: grupo.c W = 0.95855, p-value = 0.2843	En los tres grupos el p-value > 0.05 nos indica que no hay diferencia significativa con respecto a la normal.

El comando para el ANOVA es: aov()

En este caso al tener los datos en un data.frame, ejecutamos:

res.aov=aov(valor~grupo,ej1)

```
> res.aov=aov(valor~grupo,ej1)
> res.aov
Call:
   aov(formula = valor ~ grupo, data = ej1)

Terms:
                 grupo Residuals
Sum of Squares  246.0783 1859.2576
Deg. of Freedom        2        87

Residual standard error: 4.622854
Estimated effects may be unbalanced
```

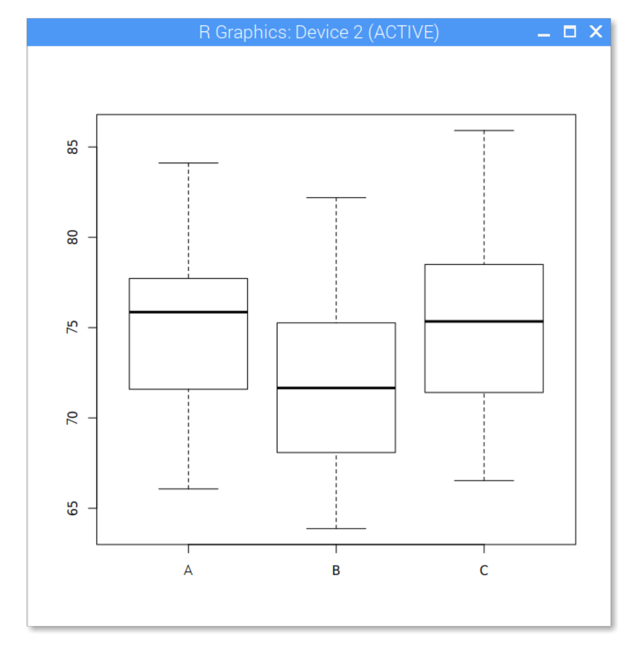
Al operar el comando summary() a los resultado obtenidos (res.aov) nos da:

```
> summary(res.aov)
            Df Sum Sq Mean Sq F value  Pr(>F)
grupo        2  246.1  123.04   5.757 0.00449 **
Residuals   87 1859.3   21.37
---
Signif. codes:  0 ‘***’ 0.001 ‘**’
0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
>
```

En este caso, se encuentra diferencia significativa entre los grupos.

Esto lo podemos apreciar a través del diagrama de caja y bigotes elaborado por grupo a través del comando:

> boxplot(ej1$valor~ej1$grupo)



Al encontrar diferencia significativa (p-value <0.05) es necesario indicar la media y desviación estándar por condición.
tapply(ej1$valor,ej1$grupo,mean) y
tapply(ej1$valor,ej1$grupo,sd)

```
> tapply(ej1$valor,ej1$grupo,mean)
       A        B        C 
75.21755 71.75839 75.31268 

> tapply(ej1$valor,ej1$grupo,sd)
       A        B        C 
4.315972 4.608699 4.923881 
```

Podemos simplificar la escritura convirtiendo las columnas a objetos con el comando attach(), en este caso:

```
attach(ej1)
```

Así solo escribimos:

tapply(valor,grupo,mean) y tapply(valor,grupo,sd). Al finalizar escribimos detach(ej1) para cancelar el attach()

```
> attach(ej1)
The following objects are masked _by_ .GlobalEnv:

    grupo, valor

The following objects are masked from ej1 (pos = 3):

    grupo, valor

> tapply(valor,grupo,mean)
       A        B        C 
75.21755 71.75839 75.31268 

> tapply(valor,grupo,sd)
       A        B        C 
4.315972 4.608699 4.923881 

> detach(ej1)
>
```

Prueba de Kruskal-Wallis, alternativa no paramétrica al ANOVA para grupos independientes. kruskal.test()

El test de Kruskal-Wallis, también conocido como test H. es la alternativa no paramétrica al ANOVA de una vía para grupos independientes.
Busca contrastar la hipótesis que "k" muestras han sido obtenidas de la misma población.

Ho: Todas las muestras provienen de la misma población (distribución).
Ha: Al menos una muestra proviene de una distribución distinta.

Condiciones:
No es necesario que las muestras provengan de una distribución normal.

- Todos los grupos deben tener varianzas similares. Se puede comprobar con representaciones gráficas o con los test de Levernne o Barttlet.
- Misma distribución para todos los grupos. No tiene porque ser normal, pero ha de ser igual en todos por ejemplo que todos muestren asimetría hacia la derecha).

Si estos requisitos se cumplen, el estadístico H del test de Kuskal-Wallis se compara con:

- Si el tamaño de grupos es igual a 3 y el número de observaciones no es mayor que 5, se recurre a tablas con valores teóricos de H.
- En el resto de casos se asume que el estadístico H sigue una distribución χ^2 con k-1 grados de libertad (siendo k el número de grupos a comparar)

Generalmente se considera que se debe emplear Kruskal-Wallis, cuando las poblaciones a comparar sean claramente asimétricas, se cumpla que todas lo sean en la misma dirección y que la varianza sea homogénea.

Si la varianza no es homogénea el test adecuado es un ANOVA con corrección de Welch.

Ejemplo:
Teniendo el grupo "Datos1", el cual está constituido por 3 grupos "A", "B" y "C" cada uno de 6 datos, con números del 1 al 6. Queremos determinar si hay diferencia entre los grupos.

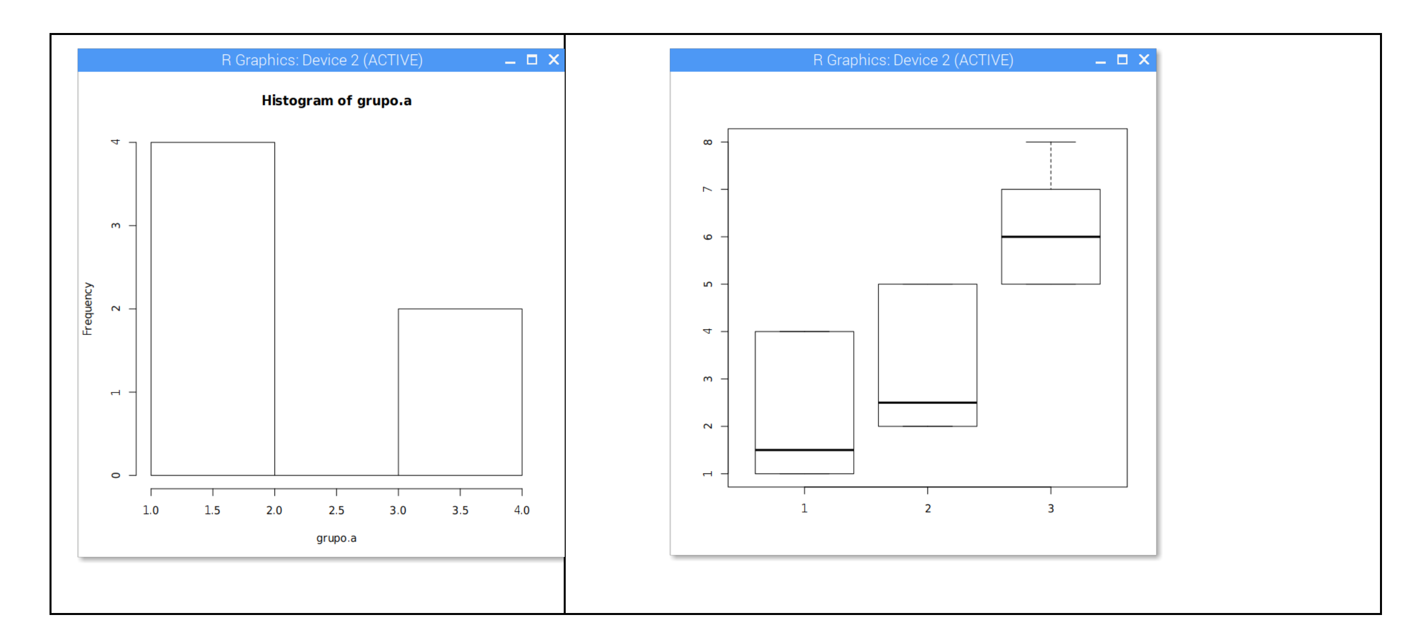
Al analizar los grupos tenemos:

```
> summary(Datos1)
 grupos     valor
 A:6    Min.   :1.000
 B:6    1st Qu.:2.000
 C:6    Median :4.000
        Mean   :3.833
        3rd Qu.:5.000
        Max.   :8.000
```

```
> grupo.a=subset(Datos1$valor,Datos1$grupos=="A")
> shapiro.test(grupo.a)

      Shapiro-Wilk normality test

data:  grupo.a
W = 0.75467, p-value = 0.02212
```



Los grupos no cumplen con la normalidad, p-value < 0.05 en el test de Shapiro.

Tienen la misma distribución, como lo demuestran sus histogramas y presentan una aparente diferencia.

Queremos determinar si la diferencia es significativa, pero no se cumplen los supuestos para la aplicación de ANOVA, por lo que se recurre a la prueba de Kruskal-Wallis

```
> kruskal.test(Datos1$valor,Datos1$grupos)

        Kruskal-Wallis rank sum test

data:  Datos1$valor and Datos1$grupos
Kruskal-Wallis chi-squared = 11.136, df = 2, p-value = 0.003818
```

Encontramos diferencia significativa, con un p-value < 0.05.

Comparaciones Post-Hoc

Al encontrarse diferencia significativa, implica que al menos dos grupos son significativamente diferentes, pero no nos indica cuales.
Para determinarlo es necesario compararlos entre ellos.

Esto podemos hacerlo por medio de “pairwise.wilcox.test()”

```
> pairwise.wilcox.test(Datos1$valor,Datos1$grupos)

        Pairwise comparisons using Wilcoxon rank sum test

data:  Datos1$valor and Datos1$grupos

        A       B
B 0.187      -
C 0.013      0.030

P value adjustment method: holm
```

Lo que nos indica una diferencia significativa del Grupo C con respecto a los otros dos, con valores de p menores a 0.05.



12. Correlación.

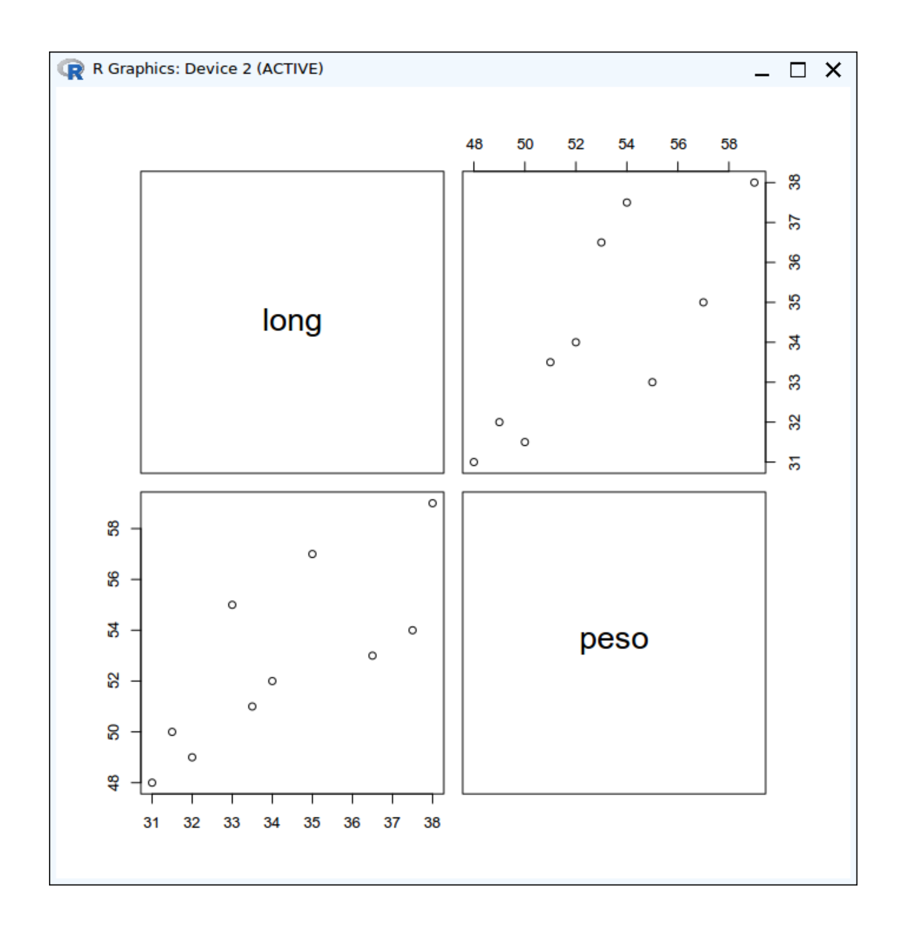
El coeficiente de correlación se puede calcular por métodos paramétricos y no paramétricos.

Coeficiente paramétrico → Coeficiente de Correlación de Pearson.
Coef. No paramétrico → Coef. De Correlación por Rangos de Spearman.

Coeficientes de Correlación.

Teniendo los datos:

```
> peso
 [1] 51 59 49 54 50 55 48 53 52 57
> long
 [1] 33.5 38.0 32.0 37.5 31.5 33.0 31.0 36.5 34.0 35.0
>
>pairs(long~peso)    # permite elaborar un plot de correlación.
```



Esta gráfica podemos generarla sin necesidad de ninguna otra librería y resulta práctica pero un poco simple.

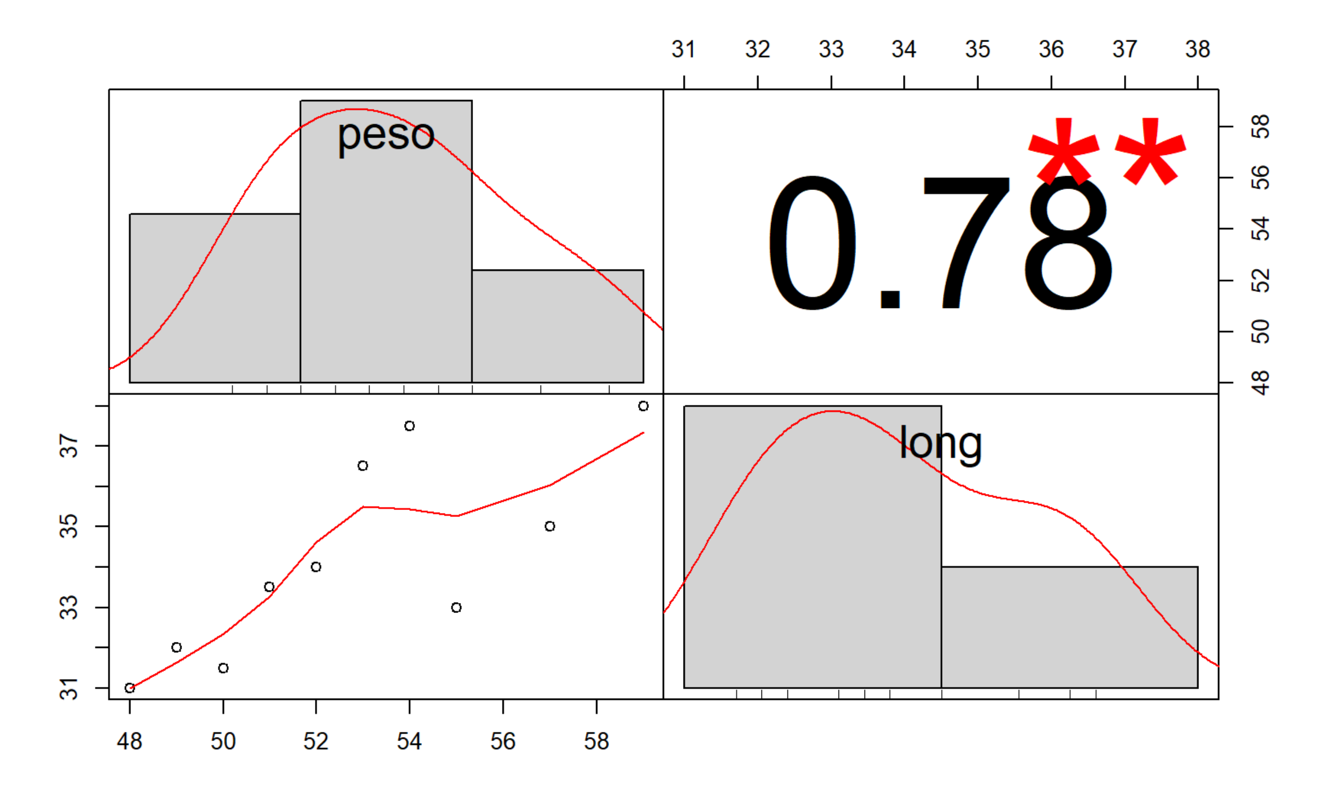
Si se van a trabajar varios análisis de correlación, existen algunos paquetes que nos dan un resultado más estético.

Una forma de obtener una gráfica de correlación de una forma elegante es a través del paquete "PerformanceAnalytics"

```
library(PerformanceAnalytics)    # Nuestros datos es mejor tenerlos en un data.frame

dat1 <- data.frame(peso, long)

chart.Correlation(dat1)
```



Esto nos permite ver el coeficiente de correlación, el comportamiento individual de cada grupo y su gráfica de correlación.

Pero no es indispensable hacerlo a través de la instalación de paquetes, podemos trabajar con el programa base sin ningún problema.

Cálculo de la correlación. cor(x,y)

Utilizando los datos presentados en la sección anterior, tenemos que para el cálculo del coeficiente de correlación de Pearson utilizamos:

```
> cor(peso,long)
[1] 0.7794691
```

Podemos utilizar también el comando "cor.test()" para calcular el coeficiente de correlación de Pearson, su intervalo de confianza y su significancia.

Asi:

```
> cor.test(peso,long)

        Pearson's product-moment correlation

data:  peso and long
t = 3.5194, df = 8, p-value = 0.007853
alternative hypothesis: true correlation is not equal to 0
95 percent confidence interval:
 0.2942560 0.9452104
sample estimates:
      cor
0.7794691
```

Correlación por rangos de Spearman. cor(x,y,method=”spearman”)

Cuando existan dudas de si se cumplen las premisas para la aplicación de Pearson, la opción es la prueba no paramétrica de Spearman.

Ejemplo:

Para dos series de datos “x” y “y” , siendo estas:

```
> x=c(44.4,45.9,41.9,53.3,44.7,44.1,40.7,45.2,60.1)
> y=c(2.6,3.1,2.5,3,3.6,4,3.2,2.8,3.8)
```

Probamos normalidad en ambos grupos, a través de la prueba de Shapiro:

para el grupo “x” tenemos:

```
> shapiro.test(x)
        Shapiro-Wilk normality test
data:  x
W = 0.80304, p-value = 0.02212    ← El grupo no es normal, (p<0.05)
```

Para el grupo ”y” tenemos:

```
> shapiro.test(y)
        Shapiro-Wilk normality test
data:  y
W = 0.94856, p-value = 0.6743    ← El grupo es normal, (p>0.05)
```

Como un grupo no cumple con la normalidad, no se puede aplicar Pearson y debemos aplicar la prueba de Spearman.

El comando es prácticamente igual que para Pearson, con el modificador "method="spearman" "

```
cor(x, y, method="spearman")


> cor(x,y,method="spearman")
[1] 0.1666667
```

De igual manera para calcular la significancia, el intervalo de confianza y el coeficiente en un solo paso, utilizamos "cor.test(x,y, method="spearman")

```
> cor.test(x,y,method="spearman")


        Spearman's rank correlation rho


data:  x and y
S = 100, p-value = 0.6777
alternative hypothesis: true rho is not equal to 0
sample estimates:
      rho
0.1666667
```



13. Análisis de regresión lineal.

El análisis de regresión se utiliza cuando el investigador sabe que existe una relación entre las variables y busca determinar la curva que mejor se ajuste a sus datos.

Para el análisis de regresión lineal, deben cumplirse los siguientes supuestos:

1. En la población, la relación entre las variables "x" e "y" debe ser aproximadamente lineal.
2. Los residuos se distribuyen según una distribución normal de media 0.
3. Los residuos son independientes.
4. Los residuos tienen una varianza poblacional constante.

Siendo:

> peso=c(61,60,78,62,66,60,54,84,68)	# creamos un vector "peso".
> altura=c(162,154,180,158,171,169,166,176,163)	# creamos un vector "altura".
> ej1=data.frame(peso,altura)	# creamos el data.frame "ej1" con la unión de "peso" y "altura"

Operamos el análisis de regresión lineal a través del comando: lm() con la siguiente sintaxis:

```
lm(variable1~variable2,data="nombre del data.frame")
```

y este resultado lo ingresamos en un vector, para poder evaluar el sumario, quedando al final de la siguiente manera:

nombre.vector=lm(variable1~variable2,data="nombre del data.frame")

En nuestro ejemplo, trabajando con el data.frame llamado ej1, escribimos:

```
> reg1=lm(peso~altura,data=ej1)
```

Luego leemos los resultados a través del comando:

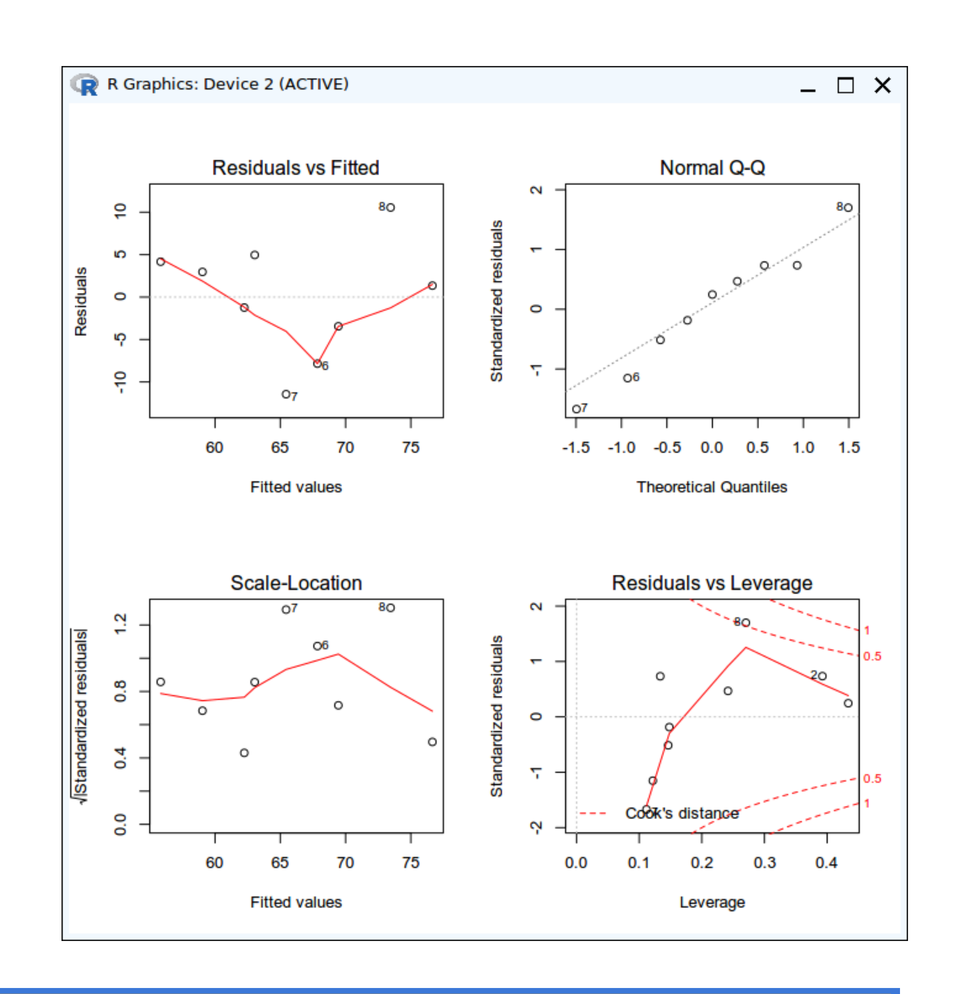
> summary(reg1)	Call: lm(formula = peso ~ altura, data = ej1) Residuals: Min 1Q Median 3Q Max -11.444 -3.447 1.347 4.164 10.549 Coefficients: Estimate Std. Error t value Pr(>|t|) (Intercept) -67.4679 51.1995 -1.318 0.229 altura 0.8007 0.3071 2.608 0.035 * --- Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1 Residual standard error: 7.268 on 7 degrees of freedom Multiple R-squared: 0.4927, Adjusted R-squared: 0.4203 F-statistic: 6.799 on 1 and 7 DF, p-value: 0.03504

Luego, operamos un cuadro de análisis de varianza sobre el resultado de regresión, para obtener la significancia.

> anova(reg1)	Analysis of Variance Table Response: peso Df Sum Sq Mean Sq F value Pr(>F) altura 1 359.15 359.15 6.7994 0.03504 * Residuals 7 369.74 52.82 --- Signif. codes: 0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Verificamos los supuestos con :

```
par(mfrow=c(2,2))
plot(reg1)
```



Y elaboramos la ecuación de predicción de la siguiente manera:

Coefficients: Estimate Std. Error t value Pr(>|t|) (Intercept) -67.4679 51.1995 -1.318 0.229 altura 0.8007 0.3071 2.608 0.035 *	Los valores estimados son: -67.4679 y 0.8007

peso = -67.4679+0.8007 * altura.



14. Referencia de comandos útiles.

Algunos comandos en R:

data()	conjuntos de datos disponibles. Se listan los ficheros de datos que suministra o incorpora el programa.
attach(fichero)	carga en memoria las variables del data frame
detach(fichero)	descarga de la memoria las variables del data frame
ls()	Lista de los objetos que están en memoria.
rm(objeto)	Elimina el objeto en memoria
attributes(data.frame)	Lista de los objetos en memoria

help.start()	Con esta ayuda en html es posible realizar búsquedas.
getwd()	Directorio de trabajo
setwd();	Cambia el directorio, como `setwd(''C:/data'')`
na.omit(x)	elimina las observaciones con datos ausentes (NA)
is.na(x)	devuelve TRUE cuando encuentra valor omitido (NA)
which(is.nan(x))	devuelve los indices de los valores con TRUE para (NA)

tapply(*variable*, *factor*, *func*)	aplica la función *func* a las variables *Var* por factores
apply(*data.frame*, op, *func*)	aplica una función a las filas si op=1 o a las columnas si **op=2**, del data.frame
table(f_1,f_2)	calcula la tabla de frecuencias para factores f_1,f_2
split (*Variable*, *Factor*)	descompone la variable según los factores

Comandos de uso básico de R:

- Instalación/carga de datos:

Nombre Comando	Explicación
install.packages()	Visualiza los paquetes de datos disponibles en internet.
install.packages(*name*)	Descarga el paquete indicado.
library()	Visualiza los paquetes disponibles.
library(*name*)	Carga el paquete indicado.
data()	Visualizar los datos disponibles.
data(*name*)	Carga en memoria el dato indicado.

- Manejo de datos:

Nombre Comando	Explicación
class(*data*)	Muestra el tipo de objeto.
dim(*data*)	Muestra las dimensiones del objeto.
ncol(*data*), nrow(*data*)	Muestra el número de columnas/filas del objeto.
names(*data*)	Muestra los nombres de las columnas.
objects(), ls()	Visualiza las variables cargadas en memoria.
rm(*data1*, *data2*)	Elimina las variables indicadas.
help(*data*), ?*data*	Muestra la ayuda asociada con el comando o variable.
Ctrl+L	Borra la pantalla.

Operadores:

Operadores					
	Aritméticos		**Comparativos**		**Lógicos**
+	Adición	==	Igual a	&	Y lógico
-	Substracción	!=	Diferente de	!	NO lógico
*	Multiplicación	<	Menor que	|	O lógico
/	División	>	Mayor que	is.na(x)	Ausente?
^	Potencia	<=	Menor o Igual que		
%/%	División Entera	>=	Mayor o Igual que		

Funciones:

Funciones			
	Matemáticas		**Estadísticas**
sqrt(x)	Raíz de x	mean(x)	Media
exp(x)	Exponencial de x	sd(x)	Cuasidesviación
log(x)	Logaritmo natural de x	var(x)	Varianza
log10(x)	Logaritmo base 10	median(x)	Mediana
length(x)	Número de elementos	quantile(x,p)	Quantiles
sum(x)	Suma los elementos de x	cor(x,y)	Correlación
prod(x)	Producto de los elementos	max(x)	El máximo
sin(x)	Seno	min(x)	El mínimo
cos(x)	Coseno	range(x)	Retorna el máximo y mínimo
tan(x)	Tangente	sort(x)	Ordena las componentes de x
round(x,n)	redondea a n dígitos	which(condición)	los índices que cumplen la condición
cumsum(x)	calcula las sumas acumuladas x_1, $x_1 + x_2$, $x_1 + x_2 + x_3$, $x_1 + x_2 + \ldots + x_n$	summary	Resumen de las variables
		choose(n, k)	número combinatorio de n sobre k

Comandos para el análisis numérico de datos:

Nombre Comando	Explicación
summary(*data*)	Resumen estadístico
min(*data*)	Mínimo
max(*data*)	Máximo
range(*data*)	Rango
mean(*data*)	Media aritmética
median(*data*)	Mediana
length(*data*)	Tamaño
sd(*data*)	Desviación típica
var(*data*), cov(*data*)	Varianza
cor(*data*)	Correlación
quantile(*data*, 0.25)	Cuantil Q1
quantile(*data*, 0.75)	Cuantil Q3
IQR(*data*)	Rango intercuartílico
sort(*data*)	Ordenar
table(*data*)	Tabla de frecuencias absolutas

Comandos para el análisis gráfico de datos:

Nombre Comando	Explicación
stem(*data*)	Diagrama de tallos y hojas
hist(*data*)	Histograma
boxplot(*data*)	Gráfico boxplot
plot(*data1*, *data2*)	Gráfico de puntos
pairs(*data*)	Gráfico de dispersión cruzado

Bibliografía.

Bibliografía

1. R Core Team, An Introduction to R,
   https://cran.r-project.org/manuals.html , Copyright © 1999–2018 R Core Team.
2. Emmanuel Paradis, R para Principiantes, Institut des Sciences de l'Évolution Universit Montpellier II, traducido por Jorge A. Ahumada RCUH/ University of Hawaii & USGS/ National Wildlife Health Center. Emmanuel Paradis (3 de marzo de 2003)
3. David Sulmont, "Manejo de datos en R" - Pontificia Universidad Católica del Perú, 26 de octubre de 2014.
   https://rstudio-pubs-static.s3.amazonaws.com/37899_6f971734dae34e978d3153cb10f4cf82.html
4. https://rparatodos.wordpress.com/2011/11/22/libros-de-r-gratuitos/
5. Instalación de R en Windows. Wikiversidad, https://es.wikiversity.org/wiki/Como_usar_R/Instalación
6. R Core Team (2015). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.



Cirujano dentista fundador de la Clínica OCR, graduado de la Universidad de San Carlos en 1993, profesor titular de la Facultad de Odontología de la Universidad de San Carlos de Guatemala desde 1995..

Profesor Coordinador del curso de Estadística por más de 25 años, ha participado en múltiples investigaciones, miembro fundador de Horizontes Universitarios y autor de varios libros.

El presente libro busca dar un enfoque sencillo y práctico al uso de R para el análisis de datos en investigación, sin profundizar en los aspectos de programación, para ser aplicado principalmente en investigaciones pequeñas, estudios de tesis, ejercicios de aprendizaje de bioestadística, resolución de ejercicios para estudiantes que cursan estadística y para el análisis de datos en el ámbito docente.